武漢大學
WUHAN UNIVERSITY

武汉大学

本科生课程讲义及知识总结

课程名称： 计算机图形学
开课学院： 遥感信息工程学院
开课时间： 2019-2020 年度第二学期

2020. 武汉大学

目录

第一章·概述

➢ 什么是计算机图形学

1. **计算机图形学的定义：**是一种使用数学算法把二维或三维图形转化为计算机显示器的栅格形式的科学。（数据可视化的科学）
2. **显示器：**是由纵横都对齐排列的像素组成。【分辨率为 1024x768 的显示器，将整个屏幕分成 768 行、1024 列的大小相同的像素阵列，每个像素都是一个矩形】
3. **数学算法：**是图形学的基本内容。表示一种单向映射关系，将图形模型映射成（变成）计算机显示器上的结果。算法的实质就是在像素阵列中选择一些像素，组成图形，将精准的数学描述形象地表示出来。

➢ 计算机图形学研究内容

1. **图形生成：**研究各种图形生成的数学模型：直线、圆、曲线、曲面、立体模型等。
2. **图形处理：**研究图形模型变换、裁剪、投影等操作。
3. **图形显示（真实感技术）：**研究图形填充、图形消隐、光照模型、颜色模型等技术方法。

➢ 计算机图形系统

1. **计算机图形系统主要由两部分组成**：硬件设备和软件系统。

   【计算机图形系统与一般计算机系统的最主要的区别：具有图形的输入、输出设备、以及必要的交互设备；对主机的运行速度、存储容量要求高】
2. **图形系统至少要包括**：计算、存储、输入、输出、对话等五项基本功能。
3. **图形系统的分类：**以大型机为基础的图形系统、以中型机或超级小型机为基础的图形系统、以工作站为基础的图形系统、以微机为基础的图形系统。
4. **图形软件标准**：
   1) 数据及文件格式标准：图形系统及其相关应用系统中各界面之间进行数据传送和通信的接口标准；
   2) 子程序界面标准：供图形应用程序调用的子程序功能及其格式标准。
5. **图形系统标准分类：**
   1) **面向图形设备的接口标准**：

a) 计算机图形元文件(CGM)，(CRT,Mouse,…)

b) **计算机图形接口(CGI).设备驱动程序**：提供控制图形硬件的一种与设备无关的方法，可看作图形设备驱动程序的一种标准，在用户程序和虚拟设备之间，以一种独立于设备的方式提供图形信息的描述和通信。

2) **面向应用软件的标准**：

a) 程序员层次交互式图形系统（PHIGS）,GL (图形程序包)

b) （三维）图形核心系统（3D-)GKS：提供了在应用程序和图形输入输出设备之间的功能接口

3) **面向图形应用系统中工程和产品数据模型及其文件格式**：

a) 基本图形转换规范（IGES）

b) 产品数据转换规范（STEP）

6. 图段

1) 定义：一组图形元素的集合，该集合成为图形操作的基本单元。

2) 作用：

a) 方便用户的增、栅、改；

b) 便于图形模块化的实现

c) 节省计算工作量

3) 性质：可变性、可见性、醒目性、可检测性、优先级可控性等。

4) 操作特性：

a) 是一个任意的二维操作；

b) 为了便于图段在不同的工作站上传送，必须设置实

c) 现图段的插入及相关的操作。

➢ 图形硬件设备——图形显示设备

1. **随机扫描方式**：出现在显像管作为图形显示设备的早期时代，只能绘制线框图。【电子束可随意移动，只扫描荧屏上要显示的部分，只画了组成图形的像素，形成图形但不能产生图像】

2. **光栅扫描方式**：需要所有的像素参与，不同的图像只是各个像素的亮点不同。一方面逐行逐列扫描每一个像素，使每个都被画到，另一方面，在每个像素被扫到时，通过

调整栅极电压的大小，确定该像素应有的亮度值。【存在栅极电压控制和 X、Y 偏转电压控制三套系统，早期电视机是光栅扫描方式】

3. **彩色显像管（属于光栅扫描方式）**：红绿蓝三套电子束发射与控制设备，每一个像素也是由红绿蓝三个子像素组成，红绿蓝三套电子枪轰击各自的子像素。
4. **数字计算机显示方式**：将光栅扫描与图像信号分开，光栅扫描是固定不变的，图像信号部分增加帧缓存器（即帧存体、显示存储器、显存)。【帧缓存器一个存储单元对应一个像素，存储单元存储了像素的颜色和亮度相关信息，只要把图像数据拷贝到帧缓存器中，就可以由显示器硬件实现图像的显示】
5. **彩色表（又称查色表）**：8 比特及其以下的图像，都有一个颜色查找表，记录了本图像的存储单元中所有取值所应该显示的 24 比特颜色值。

   【8 比特及其以下的图像，存储单元值，是颜色查找表的索引值】

   【一维线性表，其每一项的内容对应一种颜色，它的长度由帧缓存单元的位数决定，例如：每单元有 8 位，则查色表的长度为 $2^8=256$】

   【目的：在帧缓存单元的位数不增加的情况下，具有大范围内挑选颜色的能力。】
6. **新型的显示设备**：每个像素是一个发光半导体单元，像素的亮度、颜色不再用电子束轰击，而是靠电信号激发，电信号的大小由帧存体数据决定，新型显示器依然由像素阵列组成。

➢ 图形硬件设备——图形输入输出设备

1. 二维输入设备：鼠标、图形输入板、跟踪球、光笔、触摸屏、操纵杆、扫描仪。
2. 三维输入设备：空间球、数据手套。
3. 图形输出设备：
   1) 打印机：彩色喷墨打印机、激光打印机、静电打印机、热敏打印机；
   2) 绘图仪：平板绘图仪、滚筒绘图仪。

➢ 图形软件系统

1. 基本图形软件的内容：系统管理程序、图形变换、实时输入处理程序、交互处理程序。
2.

➢ 计算机图形学与其他学科的关系

3. 计算机图形学与计算机图象处理等互逆；
4. 计算机图形学是虚拟现实技术、3D 城市建模等专业课的前导基础课；
5. 与航片、卫片打交道，从图象上提取地物特征，在图象上直观地表现这些特征。

第二章·基本图形的生成

➢ 计算机图形的绘制方法以及先导知识

1. 在一个矩形的点阵阵列中选择合适的像素填入数值。
2. 选什么点：以像素中心点组成的阵列叫做点阵阵列，行列号表示（列,行）。
3. 选多少点：连通才能画出实线，连通包括 4-连通（上下左右）和 8-连通（上下左右和四个角）。一般要求满足 8-连通，选择的点要求满足：
   1) 变化平缓的图形，要求相邻像素 X（列）坐标相差为 1；
   2) 变化陡峭的图形，要求相邻像素 Y（行）坐标相差为 1。

➢ 基本图形生成

1. **图形生成算法**都使用递推法，在当前点已经确定的基础上，计算下一个点。【递推需要有初始条件和结束条件，注意误差的累计问题】
2. **图形生成算法要求**：速度快
   1) 避免使用乘除法，多用加减法；
   2) 避免使用浮点运算，多用整形运算；
   3) 避免使用函数。

➢ 直线生成方法

1. **DDA 直线生成方法**
   1) **原理**：直线上距离相同的点，x 和 y 坐标的差值也相等。
   2) **方法**：给出起点和终点坐标（一定为整数），递推加法先找出线段上的精确点，再以距离精确点最近的像素代替，绘制出直线(x0,y0)-(x1,y1)。【假定 x1>x0。如果不满足，则交换起、终点，在这个假定下，线段的绘制都是从左开始，在右边结束】

3) **流程：**

A. 判断直线是否水平、垂直线，如果是，循环语句直接画，否则进入下一步

B. 判断直线类型，对于类型 2、3、4，通过对称变换化为类型 1

C. 确定起始、终止端点，如果必要，通过交换使(x0,y0)为起始点，(x1,y1)为终止点

D. 用类型 1 的直线生成方法确定每一个点

E. 通过对称逆变换将每一个点变换到位

4) **直线(x0,y0)-(x1,y1)的 6 种情况：**

A. **x1-x0>y1-y0>0（平缓升）**：递推方法：x 方向增加 1，则 y 方向增加 m= (y1-y0) / (x1-x0)并对 y 四舍五入取整数 int(y_{i+1}+0.5)；初始条件为：(x_0,y_0)和 m；终止条件为 x_{i+1}>x1；用函数对选定的像素填充 SetPixel(x_{i+1},int(y_{i+1}+0.5), color)；

【缺点：有一个浮点运算，有一个取整函数】

B. **x1-x0>y0-y1>0（平缓降）**：对 AB 做关于 x 轴的对称变换得到直线（x0,-y0）-(x1,-y1)，再用第一种方法对直线（x0,-y0）-(x1,-y1)求出所有中间像素（x,y），绘制像素（x,-y）。

C. **y1-y0 > x1-x0>0（陡峭升）**：对 AB 做关于直线 y=x 的对称变换得到（y0, x0）-(y1, x1)，再用第一种方法对直线（y0, x0）-(y1, x1)求出所有中间像素（x,y），绘制像素（y, x）。

D. **y0-y1 > x1-x0>0（陡峭降）**：对 AB 先做关于 x 轴的对称变换，再做关于直线 y=x 的对称变换得到直线（-y0, x0）-(-y1, x1)，再用第一种方法对直线（-y0, x0）-(-y1, x1)求出所有中间像素（x,y），绘制像素（y, -x）。

E. **水平线（x0,y0）-（x1,y0）**：for(x=x0;x<=x1;x++) drawdot(x,y0);

F. **垂直线（x0,y0）-（x0,y1）**：for(y=y0;y<=y1;y++) drawdot(x0,y);

G. **程序设计基本做法**：尽量使用已有的程序模块解决新的问题。目的：提高程序的可靠性、重用性，降低工作量。

2. **中点直线算法：**

1) **原理：**

A. 如果 y_i+1>M，选 C，即 $y_{i+1,r}$=$y_{i,r}$+1；

B. 如果 y_{i+1}<=M，选 B，即 $y_{i+1,r}$=$y_{i,r}$

2) **初始条件：**：(x_0,$y_{0,r}$) = (x0,y0)，d_0 =△x-2△y

3) **递推公式：**

$$x_{i+1}=x_i+1\,;\ y_{i+1}=\begin{cases} y_{i,r}; & d_i>0 \\ y_{i,r}+1\,; & d_i\le 0 \end{cases}$$

$$d_i+1=\begin{cases} d_i-2\Delta y, d_i>0 \\ d_i-2(\Delta y-\Delta x), d_i\le 0 \end{cases}$$

4) **终止条件：** $x_i>=x_1$

5) 中点方法全部是整型数的加减法和乘 2 运算【乘 2 运算在二进制数中：左移 1 位，末位加 0】

3. **Bresenham 直线生成方法：**

1) **原理：**

A. 当 $y_{i+1}>y_{i,r}+0.5$，选 C，即 $y_{i+1,r}=y_{i,r}+1$

B. 否则选 B ，即 $y_{i+1,r}=y_{i,r}$

2) **初始条件：** $x_0=x_0, y_0=y_0$， $e_0=2\times\triangle y-\triangle x$

3) **递推公式：**

$$x_{i+1}=x_i+1\,;\ y_{i+1}=\begin{cases} y_{i,r}+1; & e_i>0 \\ y_{i,r}; & e_i\le 0 \end{cases}$$

$$e_i+1=\begin{cases} e_i+2\Delta y-2\Delta x, e_i>0 \\ e_i+2\Delta y, e_i\le 0 \end{cases}$$

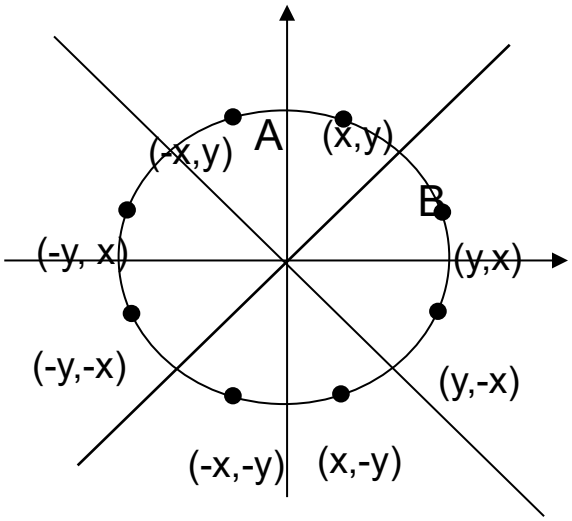
4) **终止条件：** $x_i>=x_1$

➢ 圆的生成

1. **生成圆的 Bresenham 方法：**

1) 圆心在原点：只要画出 1/8 弧段（AB 段），根据对称关系就能画出整个圆。

2) **原理：** 在第 i 点已经选择 A 时,第 i+1 点只能选择 B 或 C，B 或 C 哪个离圆近就选哪个。设圆心为 O，半径为 R，比较|OB-R|与|OC-R|大小，前者大选 C，后者大选 B。为了减少乘方开方运算，比较$|OB^2-R^2|$与$|OC^2-R^2|$大小【精度换速度】



3) **判别式：** $d_i=((x_i+1)^2+y_i^2-R^2)+((x_i+1)^2+(y_i-1)^2-R^2)$

4) **初始条件：** $x_0=0, y_0=R, d_0=3-2R$

5) **递推公式：**

$$x_{i+1}=x_i+1\,;\ y_{i+1}=\begin{cases} y_i-1; & d_i>0 \\ y_i; & d_i\le 0 \end{cases}$$

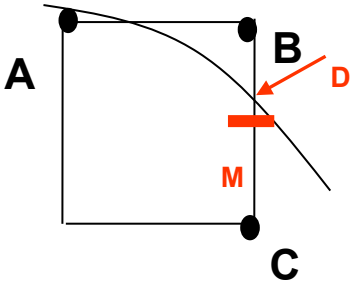
$$d_i+1=\begin{cases}d_i+4(x_i-y_i)+10, d_i>0\\ d_i+4x_i+6, d_i\le 0\end{cases}$$

5) **结束条件：** $x_i >= y_i$

6) **圆心为任意点的情况下：** 在圆心为（0，0）的圆上运营算法计算出（x_i,y_i），再画点（x_i+x_c, y_i+y_c)即可。

2. **中点圆算法：**

1) **原理：** D 为圆弧与直线 BC 的交点，M 为线段 BC 的中点，M 在圆内选 B，M 在圆外选 C。设圆的方程为 F(x,y)=0，M 在圆内时 F(M)<0，M 在圆外时 F(M)>0，M 在圆上时 F(M)=0。



2) **判别式：** d_i=F(M)

3) **初始条件：** (x_0,$y_{0,r}$)=(0,R)，d_0=F(x_0 +1,$y_{0,r}$ − 0.5)=5/4-R

4) **递推公式：**

$$x_{i+1}=x_i+1\ ;\ y_{i+1,r}=\begin{cases}y_{i,r}-1;\ d_i>0\\ y_{i,r};\ d_i\le 0\end{cases}$$

$$d_i+1=\begin{cases}d_i+2(x_i-y_{i,r})+5, d_i>0\\ d_i+2x_i+3, d_i\le 0\end{cases}$$

5) **结束条件：** **xi>=yi**

3. **生成圆的正负法**

1) **原理：** 从圆上一点出发，寻找下一个靠近圆弧的点。当前点在圆上或圆外，向圆内走一步；当前点在圆内，向圆外走一步

2) **初始条件：** x_0=x_c, y_0=y_c+R, F(x_0,y_0)=0

3) **递推公式**:

当$F(x_i,y_i)<0$ *时:*

$$x_{i+1}=x_i+1\ ;\ y_{i+1}=y_i\ ;F(x_{i+1},y_{i+1})=F(x_i,y_i)+2(x_i-x_c)+1$$

当$F(x_i,y_i)\ge 0$ *时:*

$$x_{i+1}=x_i\ ;\ y_{i+1}=y_i-1\ ;F(x_{i+1},y_{i+1})=F(x_i,y_i)-2(y_i-y_c)+1$$

4) **结束条件：** y_i<=y_c

5) **特点：** 正负法生成的圆是 4-连通的，不如前法的圆光滑。

➢ 线宽与线型的处理

1. **直线的线宽处理——线刷子**

1) 水平线刷子：对接近垂直的线用水平线刷子；

2) 垂直线刷子：对接近水平的线用垂直线刷子

3) 特点：

A. 用线刷子画的线的宽度比定义的（想得到的）线宽略窄；

B. 优点：实现简单；

C. 缺点：线段两端要么为水平的，要么是竖直的；当折线的两根直线的斜率分别大于 1 和小于 1 时，在顶点处有缺口；对于宽度为偶数像素的直线会产生偏移。

2. **直线的线宽处理——方刷子**

1) 将方形刷子的中心放在直线的端点，方形刷子的中心沿着直线移动，直到直线的另一个端点，画出周边的 n×n 个像素点。

2) 特点：

A. 对接近水平和接近垂直的线均适用，折线处没有缺口；

B. 用方形刷子画的线的宽度比定义的（想得到的）线宽略宽；

C. 缺点：线段两端要么为水平的，要么是竖直的；对于宽度为偶数像素的直线会产生偏移。

3. **直线的线宽处理——填充法**

4. **直线的线型处理：**

用一个布尔值系列来存放线型值，用一个位串来表示线型出现的周期，即 1 画 0 不画

第三章 · 二维图形的填充

➢ 区域填充

1. 二维图形填充所要做的就是将封闭图形内部所有的点（内点）用指定的颜色表示。

2. **内点：**封闭多边形内部的点。

3. **多边形的两种表示方法**：

1) 顶点表示法，即用多边形的顶点序列来表示；

2) 点阵表示法，即用多边形的全体内点来表示。

4. **扫描转换**：图形的顶点表示转化为点阵表示。

1) 多边形扫描转换又称区域填充；

2) 常用方法：①逐点判断方法；②扫描线算法；③边缘填充算法

5. **逐点判断方法**：对于屏幕上的每一个像素判断是否内点

1) **内点判断方法**：①射线法；②累计角度法；③编码方法

2) **射线法**：

A. 原理：从一点出发的射线，若与多边形边界的交点个数为奇数，为内点；偶数，不是内点；

B. 射线方向定为向右的水平线。这样只要计算射线与边界线的交点，或从待判定点出发，水平向右找边界点，直到显示器边缘。

C. 缺点：计算量过大：①要计算射线与每条边的交点；②要判断交点是否在该点的右边；③要判断交点是否在边的两个端点之间。

➢ 区域填充的扫描线算法

1. **扫描线算法**：充分利用了像素之间的各种关系，即区域的连贯性、扫描线的连贯性和边的连贯性，减少了计算量，提高了速度。

1) **区域的连贯性**：两条扫描线之间的区域分割成若干个相邻区域，两个相邻区域分别分属多边形内外，只要确定这些区域中任何一个区域内的一个点与多边形的内外关系，就可确定所有这些区域的内外关系。

2) **扫描线的连贯性**：一条扫描线与多边形非水平边的交点数为偶数，这些交点从左到右排序，奇数交点与其后的偶数交点之间所有的点都是多边形内点。

3) **边的连贯性**：相邻扫描线与同一条边的交点坐标在垂直方向相差 1，在水平方向相差 d, d 等于边的斜率的倒数。

4) 扫描线与两个非水平边的连接处的交点均为两个重合点，即理论上应有两个交点

——解决方法：不考虑与上端点相交的扫描线，即去掉每条线的上端点所对应的扫描线。

2. **扫描线算法的实现**：

1) **非水平边的数据结构**：

y_{max}	x	dx	next

其中：

y_{max}:边的上端点的 y 坐标

x:边的下端点的 x 坐标

dx:边的斜率的倒数（△x/△y）

next:指向下一条边的指针

2) **建立边表（ET）**:

A. ET 表是按边下端点的 y 坐标对非水平边进行分类的指针数组;

B. 下端点的 y 坐标为 i 的边归入第 i 类，有多少条扫描线，就设多少类边;

C. 同一类中，各边按 x 值递增的顺序排列成行，x 值相等时就按 dx 值递增的顺序排列。

D. 第一步需要要去掉水平边，即 dx 无穷大【垂直边为 0】。

3) **活化边列表（AEL）**：与扫描线相交的边结构组成，从最底下开始 y 递增，取出对应的边表，每往上取值需要注意去掉达到上端点的边，同时 x 需要变化 dx。

A. 相当于一个指向边表下一结构的指针;

B. 每次填充注意从 $X_{左}$ 填充至 $X_{右}$，左边填充最大为不小于 $X_{左}$ 的整数，右边填充最大为小于 $X_{右}$ 的整数，即为一个半开半闭区间;【填左不填右】

C. 图形中的平行线（不是平行边）：填下不填上;

4) **扫描线具体算法步骤总结**:

A. 建立 ET 表，按边下端点的 y 坐标对非水平边进行分类并排序;

B. 将扫描线纵坐标 y 的初值置为 ET 表非空元素的最小序号

C. 置 AEL 为空

D. 执行下列步骤，直到 ET、AEL 都为空:

① 如果 ET 中的第 y 类非空，将其中所有的边取出并插入 AEL;

② 如果有新的插入 AEL,则对 AEL 各边排序

③ 对 AEL 中的边两两配对，每对边中的 x 取整，获得有效填充区段，并填充

④ 将当前扫描线纵坐标 y 值递增 1

⑤ 将 AEL 中满足 y=ymax 的边删除

⑥ 对 AEL 中剩下的边的 x 字段作 x=x+dx 操作

➢ 区域填充的边缘填充算法

1. 边缘填充算法不需要频繁排序，用异或的方式绕开了排序的问题。
2. 异或的特点：同一像素以异或方式画两次，颜色擦除（即还原成处理前）。
3. **边缘填充算法**：从边上一点画到最右边，内点区域被画了奇数次，非内点区域被画了偶数次。
4. **边缘填充算法的缺陷**：
   1) 填充的颜色与指定的颜色不同；
   2) 虽然每个边缘点的操作方法一样有利于编程，但要处理大量像素，速度比扫描线算法慢。【加速方法：确定图形边缘框，只画到边缘框右边】

➢ 区域填充的种子填充算法

1. 属于面着色的另一类算法，**有两个要求**：
   1) 要求区域有一个连通的边界；
   2) 需要一颗种子点
2. 种子填充算法的适合场景：
   1) 适合于已经画出大量边界线的场合；
   2) 适合于人机交互操作方式，或者事先保留了种子点
3. **扫描线种子填充算法的过程**：
   1) 将一个种子 1 压入堆栈
   2) 从堆栈中弹出种子 1，左右填充直到边界
   3) 填充线上下搜索未填充区，每区选一粒种子 2，3，4 压入堆栈
   4) 重复直至堆栈中没有种子为止。

➢ 图案填充

1. 确定一个内点，不再用确定颜色的填充。该内点填什么颜色，取决于其在图案中的对应颜色。
2. **图案填充的步骤**：
   1) 将图案像素值存入一个$U \times V$数组
   2) 用前述各种方法计算需要填充的像素坐标(x, y)
   3) 求余运算：$u = x \div U$, $v = y \div V$
   4) 从图案数组中取出(u, v)像素值，填入像素(x, y)

3. **两种填充方案的选择**:

1) 像素坐标取设备坐标，(x,y)原点在屏幕左下（上）角；

2) 像素坐标取图形相对坐标， (x,y)原点在图形区域左下（上）角。

A. 像素坐标(x,y)首先转化成相对坐标$(x1,y1)$，$(x1,y1)$原点在图形区域左下（上）角；

B. 方案 2 多转换一次坐标，填充的图形，当图形移动后，填充图案不会发生变化。

第四章 · 二维裁剪

➢ 二维裁剪的相关概念

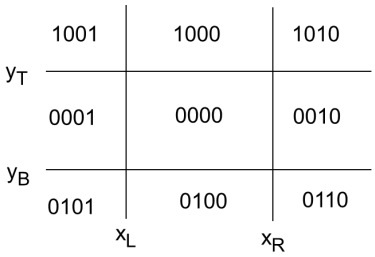
1. **裁剪窗口：** 用来指定图形显示内容的矩形区域称为裁剪窗口。在窗口内的图形被显示，窗口之外的图形被裁剪掉。
2. 二维裁剪分为直线裁剪，多边形裁剪。

➢ 常见的直线裁剪方法——Cohen-Sutherland 算法

1. 原理：

1) 平面分成 9 个区域，各区域用 4 位二进制码$C_TC_BC_RC_L$表示

2) 根据点所在区域赋予区域码，编码方法如下：



A. $y \le y_T$（窗口这边）C_T=0 否则 C_T=1

B. $y > y_B$ （窗口这边）C_B=0 否则 C_B=1

C. $x \le x_R$（窗口这边）C_R=0 否则 C_R=1

D. $x > x_L$（窗口这边）C_L=0 否则 C_L=1

E. 0000 区域为显示窗口，只有在该区域的直线段才显示。

3) 对线段进行操作：

A. **端点两码均为 0000**：可见，显示；

B. **端点两码相与结果不为 0000**：不可见，不显示；

C. **端点两码不全为零，相与结果为 0000**：不确定，可能部分可见，继续判断：

a) 用编码中，1 所对应的边与直线求交点，将线段分成两段；

b) 为两个新线段端点编码，窗口这边线段的交点，该位编码为 0，非窗口这边线段的交点，该位编码为 1；

c) 用上述步骤 1，2，3 对两个新线段进行判断，直到线段所有部分都明确可见或不可见为止。

➢ 常见的直线裁剪方法——中点分割法

1. **内侧空间与外侧空间：**【针对于某根线】

   内测空间：窗口所在的区域；

   外侧空间：窗口所不在的区域。

   ☆窗口区域是四个内侧空间的交集

2. 如果某条线段的两个端点都在同一窗口边的外侧空间，那么这个线段就在外侧空间。

   ☆窗口在外侧空间用函数表示为：LineInOutspace(x0,y0,x1,y1)==true

3. 外侧空间的线段一定在窗口外，在窗口外的线段不一定在外侧空间。

4. **中点分割算法的原理：**从起点 P_0 出发寻找最近的可见点 A，从终点 P_1 出发寻找最近的可见点 B，AB 即为分割结果。

5. **从起点 P_0 出发寻找最近可见点 A 的方法：**

   1) 用中点 P_m 将线段分隔成两段；

   2) 对前段进行分析：如果前段在外侧空间则扔掉前段，用后段代替整个线段（$p_0=p_m$）；

   3) 如果前段不在外侧空间则扔掉后段（$p_1=p_m$）；

   4) 一次去掉一半的线段，直到起点和终点相邻的时候确定 A 点（p_0 或 p_1 都行）。

6. **算法步骤：从 p0 出发寻找最近可见点 p**

   1) 如果 p0p1 不可见(LineInWindow()==false)，结束；否则：

   2) 若 p0 在窗口内(PointInWindow()==true)，不用找了，p = p0；

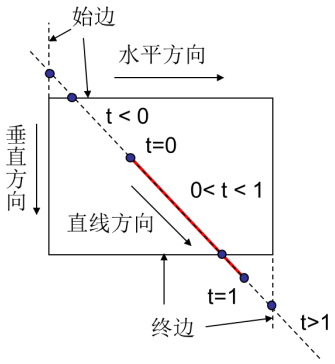
   3) 若 p0 在窗口外，取中点 pm=(p0+p1)/2；

   4) pm，p1 是否相邻？是，p= pm，结束；不是，向下走；

   5) pmp0 是否不可见？是, p0= pm ，回到 1；不是, p1= pm ，回到 1。

➢ 常见的直线裁剪方法——梁友栋-Barsky 算法

1. 算法原理：



1) 线段参数化，起点 t=0，终点 t=1。沿着直线变化方向，t 增加；

☆0-1 为线段有效区域

2) 根据直线变化方向确定两条始边，两条终边；

3) 求出直线与两条始边、两条终边的交点，记录对应的参数值；

4) 起点与始边交点的参数比较，取出一个大的参数；终点与终边交点参数比较，取出一个小的参数；

5) 两个参数对应的点为裁剪直线的起点与终点。

2. **具体计算方法：**

1) 用参数方程表示直线(x0,y0)—(x1,y1)：$x = x0 + (x1 - x0)t;\ y = y0 + (y1 - y0)t$

2) 根据直线走向，确定窗口边的始边与终边；

3) 始边终边参数直接代入参数方程计算直线与始边的交点参数，记为$t0', t0''$；计算直线与终边的交点参数，记为$t1', t1''$

4) 比较大小:$t0 = max(0, t0', t0'')\ t1 = min(1, t1', t1'')$

5) 如果$t0 \geq t1$，直线不可见。否则计算 t0，t1 对应的直线点(x0, y0), (x1, y1)，并显示

6) 当线段为水平或垂直线时，即 y1=y0 或 x1=x0，先判断$y1 > yT$ 或 $y1 < yB$或 $x1 > xR$ 或 $x1 < xL$ (该线段是否不可见)，如果不能确定，就计算交点参数。此时，只要计算一条始边和一条终边。

☆答题时需要注意确定起点和终点，还需要考虑$\Delta x = 0$ 和 $\Delta y = 0$ 的情况

➢ 多边形裁剪

1. 多边形裁剪不同于直线裁剪，多边形表示一个用多边形的边围成的一个区域，裁剪结果仍然是一个区域。部分区域的边界可能由窗口边界代替。

2. Sutherland-Hodgman 算法【矩形窗口裁剪多边形】:

1) 将多边形所有顶点依次输入到输入数组 P[ ],即$P[\ ] = P0P1P2 \ldots\ldots Pi \ldots\ldots PnP0$

☆相邻的两个顶点 PiPi+1 是多边形的一条边

☆起点 P0 也作为最后一个顶点输入到数组，以构成最后一条边 PnP0

2) 准备一个输出数组 Q[]，用来接收输出的顶点。

3) 依次从 P[]中取出一条边，前一个点放入前点变量 S，后一个点放入后点变量 P。

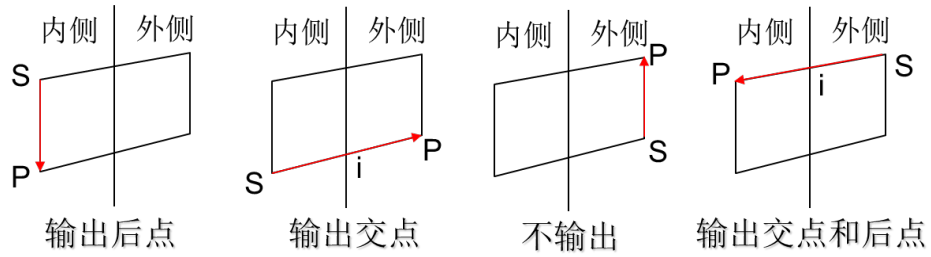
4) 按照下列原则将相应的点依次存入 Q[]:

A. 内内，输出后点 P;

B. 内外，输出交点 I;

C. 外外，不输出;

D. 外内，输出交点 I 和后点 P。



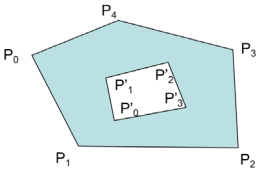
5) 返回 3，直到所有的边处理完毕。

3. **Weiler-Atherton 算法【多边形裁剪多边形】**:

1) 两项约定:

A. 被裁剪多边形称为主多边形，裁剪窗口称为裁剪多边形。

B. 约定多边形外部边界的顶点顺序取逆时针方向，内部边界的顶点顺序取顺时针方向。保持多边形区域位于有向边左侧。



C. 进点与出点:

a) 进点：主多边形边界由此进入裁剪多边形区域的交点;

b) 出点：主多边形边界由此离开裁剪多边形区域的交点

2) 步骤:

A. 建立顶点表：主多边形 A 为：$P0P1P2 \dots P0$; 裁剪多边形 B 为:$Q0Q1Q2 \dots Q0$

B. 求出主多边形和裁剪多边形的交点，并将交点按照顺序插入到两个多边形顶点表中。

C. 在两个表的相同交点之间建立双向指针

D. 利用这两个表进行裁剪:

a) 建立空的裁剪结果多边形的顶点表;

b) 选取任意一个没有被跟踪的交点为起点，将其输出到结果多边形顶点表中；

c) 如果该交点为进点，跟踪主多边形边界顶点表；为出点，跟踪裁剪多边形边界顶点表；

d) 跟踪多边形边界，每遇到多边形顶点，将其输出到结果多边形顶点表中，直到遇到新的交点；

e) 重复如此，直到回到起点；

f) 查找没有被跟踪过的交点，再进行新的跟踪，直到所有的交点都被跟踪过，裁剪结束。

➢ 字符裁剪

1. 计算机字体有点阵字体和矢量字体两种。
   1) 点阵字体优点是显示速度快，不像矢量字体需要计算；其最大的缺点是不能放大，一旦放大后就会发现文字边缘的锯齿；
   2) 矢量可以随意缩放、旋转而不必担心会出现锯齿字体方程的复杂，所以在屏幕上渲染的时候，花费的时间多。
2. **字符串裁剪的三种类型**：
   1) **串精确裁剪**：当字符串中每一个字符的方框（边界盒）都在窗口内，予以显示字符串；否则不显示字符串；
   2) **字符精确裁剪**：当字符串的某个字符方框在窗口内时，显示该字符，否则不显示该字符；
   3) **笔划/象素精确裁剪**：具体判断字符串中各字符的哪些象素、笔划的哪一部分在窗口内，处理方法同字符裁剪。

第五章·图形变换

1. 图形变换是根据模型原有的位置信息和变化的条件，计算出变换后的位置，并加以显示。

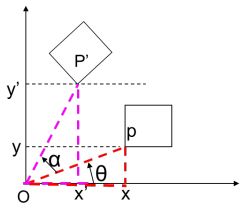
➢ 二维图形平移、旋转、缩放、对称变换、

1. 平移前的坐标：$P(x,y)$，平移后的新坐标：$P'(x',y')$，则：

   $x' = x + tx;\ y' = y + ty \rightarrow [x',y'] = [x,y]\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} + [tx,ty]$

2. 二维图形旋转是将图形绕原点旋转。

3. 图形上任意一点$P(x,y)$旋转后的位置为$P'(x',y')$。若$|OP| = R$



   $x = Rcos\theta, y = Rsin\theta$

   $x' = Rcos(\theta+\alpha) = Rcos\theta cos\alpha - Rsin\theta sin = x\ cos\alpha - y\ sin\alpha$

   $y' = Rsin\ (\theta+\alpha) = x\ sin\alpha + y\ cos\alpha$

   $[x',y'] = [x,y]\begin{bmatrix}cos\ \alpha & sin\ \alpha\\ -sin\ \alpha & cos\ \alpha\end{bmatrix}$

4. 二维图形缩放是以坐标系原点为基准将图形上任意一点$P(x,y)$在 x 轴方向缩放s_x倍，y 轴方向缩放s_y倍。

   $x' = s_x \times x \quad y' = s_y \times y \rightarrow [x',y'] = [x,y]\begin{bmatrix}s_x & 0\\ 0 & s_y\end{bmatrix}$

5. 对称变换：

   1) 关于 x 轴的对称变换。特点：$(x,y) \rightarrow (x,-y)$，变化矩阵：$\begin{bmatrix}1 & 0 & 0\\ 0 & -1 & 0\\ 0 & 0 & 1\end{bmatrix}$

   2) 关于 y 轴的对称变换。特点：$(x,y) \rightarrow (-x,y)$，变化矩阵：$\begin{bmatrix}-1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}$

   3) 关于 y=x 的对称变换。特点：$(x,y) \rightarrow (y,x)$

   4) 关于任意直线的对称变换

      A. 移动坐标，使坐标原点在直线上

      B. 旋转坐标，使 x 轴与直线重合

      C. 对图形以 x 轴做对称变换

      D. 旋转坐标，使 x 轴回到原方向

      E. 移动坐标，使坐标回到原位置

6. 错切变换：保持图形上各点的某一坐标值不变，而另一坐标值关于该坐标值呈线性变化。坐标保持不变的那个坐标轴称为依赖轴，余下的坐标轴称为方向轴

   【错切变换不仅改变图形的形状，而且改变图形的方位，还可能使图形发生畸变】

   1) **以 y 轴为依赖轴的错切变换**：$SH_y(sh_x) = \begin{bmatrix}1 & 0 & 0\\ sh_x & 1 & 0\\ 0 & 0 & 1\end{bmatrix}$

2) **以 x 轴为依赖轴的错切变换**：$SH_x\left(sh_y\right)=\begin{bmatrix}1 & sh_y & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}$

3) **以任意直线为参考轴**：要事先通过变换，将任意线与 x 轴或 y 轴重合，再按上述方法进行变换，再进行变换将任意线恢复原状：

7. **仿射变换**

1) 仿射变换的特点是变换前的平行线在变换后依然平行。

2) 仿射变换是一种线性变换，即新的坐标值是由老坐标值通过线性组合得到的。前面提到的变换都是仿射变换。

3) 仿射变换的关系式与变化矩阵：

$$\begin{cases}x` = ax+by+e\\ y` = cx+dy+f\end{cases};\ A_F=\begin{bmatrix}a & c & 0\\ b & d & 0\\ e & f & 1\end{bmatrix}$$

➢ 齐次坐标与二维变换的矩阵表示

1. (x,y)用齐次坐标(x,y,h)表示：因为第 3 维 h，在这里仅仅是为了增加一个维数，没有其他实际意义，用 1 代替。即一个二维平面点(x,y)用齐次坐标$(x,y,1)$表示。

✧ 以下的矩阵表示均为固定坐标系模式下的表示形式

2. 平移：$[x'\quad y'\quad 1]=[x\quad y\quad 1]\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ t_x & t_y & 1\end{bmatrix}$

3. 旋转：$[x'\quad y'\quad 1]=[x\quad y\quad 1]\begin{bmatrix}\cos\alpha & \sin\alpha & 0\\ -\sin\alpha & \cos\alpha & 0\\ 0 & 0 & 1\end{bmatrix}$ 【逆时针α为正】

4. 缩放：$[x'\quad y'\quad 1]=[x\quad y\quad 1]\begin{bmatrix}s_x & 0 & 0\\ 0 & s_y & 0\\ 0 & 0 & 1\end{bmatrix}$

5. 图形变换可以看成一个矩阵 T：$[x'\quad y'\quad 1]=[x\quad y\quad 1]\times T$

6. 复合变换

1) 绕任意点旋转：①平移图形，使任意点与原点重合；②绕原点（为平移后的任意点）旋转；③平移图形，使任意点回到原处。

$$T=T_1T_2T_3=\begin{bmatrix}\cos\alpha & \sin\alpha & 0\\ -\sin\alpha & \cos\alpha & 0\\ x_0(1-\cos\alpha) & -x_0\sin\alpha-y_0\cos\alpha+y_0 & 1\end{bmatrix}$$

2) 以任意点为参考点的缩放：①平移图形，使任意点与原点重合；②以原点（为平

移后的任意点）为参考点缩放；③平移图形，使任意点回到原处。

➢ 变换模式

1. **两种变换模式**：固定坐标系模式，活动坐标系模式。
2. **固定坐标系模式**：坐标系不变、图形变动。变换关系是指图形变动前后之间的坐标关系
3. **活动坐标系模式**：图形不变，坐标系变动。变换关系是指图形在新旧坐标系中的坐标之间的关系。
4. 坐标系是一种定位基准（包括原点和坐标轴方向），在基准中能够用数据（坐标值）唯一地确定目标的位置。图形坐标是相对于一种坐标系的位置定位，图形坐标是相对于该基准的测量距离。
5. 活动模式与固定模式实质一样，即两种变换关系完全等价，但变换参数相反。

活动坐标系模式		固定坐标系模式	
平移:	tx, ty	平移	$-tx, -ty$
旋转	α	旋转	$-\alpha$
缩放	Sx, Sy	缩放	$1/Sx, 1/Sy$

例如：将坐标系平移描述为：$[x' \quad y' \quad 1] = [x \quad y \quad 1]\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$

那么对应到点在固定坐标系中的变换应该表示为：

$$[x' \quad y' \quad 1] = [x \quad y \quad 1]\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -t_x & -t_y & 1 \end{bmatrix}$$

将坐标轴沿 y 轴平移-2 个单位，点的变化矩阵应该表示为：$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -(-2) & 1 \end{bmatrix}$

➢ 二维图形的显示及三维几何变换

1. **世界坐标系：**如果这个坐标是通用的、标准的，如大地坐标系，则这个坐标系称为世界坐标系。(WC)

【世界坐标系可以是二维的也可以是三维的；有右手坐标系和左手坐标系之分；各坐标轴的取值范围是整个实数域；是与设备无关的坐标系】

2. **用户坐标系**：如果这个坐标是用户按照自己熟悉或方便的方式建立的，则称为用户坐标系

3. **设备坐标系**：图形系统对图形进行输出的坐标系，称为显示坐标系或设备坐标系(DC)。

   【设备坐标系是二维的；数据类型只能是整型，取值范围受设备有效幅面限制；坐标原点因设备而异】

4. **规格化设备坐标系**：为了方便在任何尺寸的输出设备上输出图形，将设备坐标系作归一化处理，即在 x,y 方向上的最小值均为 0，最大值均为 1。这种坐标系叫归一化设备坐标系(NDC)。

   【介于世界坐标系和设备坐标系之间，与设备无关；约定坐标轴的取值范围 0 到 1】

5. **视区**：通常情况下，并不是整个屏幕都用来显示图形。往往在屏幕上划定一个平行于设备坐标轴的矩形区域作为图形显示区。这个区域称为视区。

6. **三维平移变换**：$T(t_x,t_y,t_z)=\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\t_x&t_y&t_z&1\end{bmatrix}$

7. **三维缩放**：$S(s_x,s_y,s_z)=\begin{bmatrix}s_x&0&0&0\\0&s_y&0&0\\0&0&s_z&0\\0&0&0&1\end{bmatrix}$

8. **三维旋转：绕 x 轴，y 轴，z 轴的（坐标系旋转顺时针左手螺旋为正）旋转矩阵如下：**

$$R_x(\theta)=\begin{bmatrix}1&0&0&0\\0&\cos\theta&\sin\theta&0\\0&-\sin\theta&\cos\theta&0\\0&0&0&1\end{bmatrix}\quad R_y(\theta)=\begin{bmatrix}\cos\theta&0&-\sin\theta&0\\0&1&0&0\\\sin\theta&0&\cos\theta&0\\0&0&0&1\end{bmatrix}$$

$$R_z(\theta)=\begin{bmatrix}\cos\theta&\sin\theta&0&0\\-\sin\theta&\cos\theta&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$$

☆绕任意轴旋转的过程：要事先通过变换，将任意线与 z 轴重合，再按绕 z 轴的旋转方法进行变换，再进行变换将任意线恢复原状。

$[x'\quad y'\quad z'\quad 1]=[x\quad y\quad z\quad 1]\times T_R$

$[x''\quad y''\quad z''\quad 1]=[x'\quad y'\quad z'\quad 1]\times R_z(\theta)$

$[x_1\quad y_1\quad z_1\quad 1]=[x''\quad y''\quad z''\quad 1]\times {T_R}^{-1}$

9. **错切变换：**例如以 z 轴为依赖轴，则x,y坐标依 z 坐标呈线性变换

$$SH_z(sh_x,sh_y)=\begin{bmatrix}1&0&0&0\\0&1&0&0\\sh_x&sh_y&1&0\\0&0&0&1\end{bmatrix}$$

10. **对称变换：**对称单元是平面，针对任意平面的变换：要事先通过变换，将任意平面与xoy平面重合，再按以xoy平面的对称方法进行变换，再进行变换将任意平面恢复原状

$$[x'\quad y'\quad z'\quad 1]=[x\quad y\quad z\quad 1]\times T_R$$

$$[x''\quad y''\quad z''\quad 1]=[x'\quad y'\quad z'\quad 1]\times SY_{xy}$$

$$[x_1\quad y_1\quad z_1\quad 1]=[x''\quad y''\quad z''\quad 1]\times {T_R}^{-1}$$

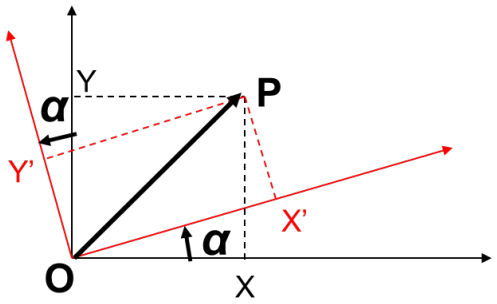
➢ 坐标系之间的变换

1. **线性代数知识：**

   1) 空间点（一个向量）在一个坐标系中的坐标等于该向量在坐标轴上的投影

   2) A 向量在 B 向量上的投影值等于 A 向量与 B 向量的单位向量之间的内积（点乘）

   $AB=|A||B|cos\boldsymbol{\alpha}$



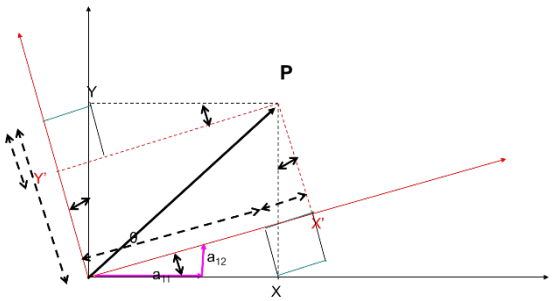
2. **坐标轴在原点的情况：**

$$[\boldsymbol{x'}\quad \boldsymbol{y'}\quad \boldsymbol{1}]=[\boldsymbol{x}\quad \boldsymbol{y}\quad \boldsymbol{1}]T_R$$

$$=[\boldsymbol{x}\quad \boldsymbol{y}\quad \boldsymbol{1}]\begin{bmatrix}\boldsymbol{a_{11}}&\boldsymbol{a_{21}}&\boldsymbol{0}\\\boldsymbol{a_{12}}&\boldsymbol{a_{22}}&\boldsymbol{0}\\\boldsymbol{0}&\boldsymbol{0}&\boldsymbol{1}\end{bmatrix}$$

$$[\boldsymbol{x}\quad \boldsymbol{y}\quad \boldsymbol{1}]=[\boldsymbol{x'}\quad \boldsymbol{y'}\quad \boldsymbol{1}]T_R^{-1}$$

$$=[x'\quad y'\quad 1]\begin{bmatrix}a_{11}&a_{12}&0\\a_{21}&a_{22}&0\\0&0&1\end{bmatrix}$$



$a11=a22=cos\alpha$

$a12=-a21=sin\alpha$

☆T_R: 正交矩阵：性质：$T_R^{-1}=T_R^T$

X 轴的单位向量为 i，Y 轴的单位向量为 j

X'轴的单位方向矢量为$(a11,a12)$，可表示为：$a11i+a12j$;

Y'轴的单位方向矢量为$(a21,a22)$，可表示为：$a21i+a22j$;

点$P(x,y)$可用矢量表示为：$x\,i+\,y\,j$

点$P(x,y)$在X'轴上的投影可用点乘得到：

$x' = (x\,i+\,y\,j)(a11i + a12j) = a11\,x\, + a12y$

$y' = (x\,i+\,y\,j)(a21i + a22j) = a21\,x\, + a22y$

3. **坐标系原点不重合：**

1) $[x' \quad y' \quad 1]=[x \quad y \quad 1]T(-x_0,-y_0)\begin{bmatrix}a_{11} & a_{21} & 0\\ a_{12} & a_{22} & 0\\ 0 & 0 & 1\end{bmatrix}$

2) $[x \quad y \quad 1]=[x' \quad y' \quad 1]\begin{bmatrix}a_{11} & a_{12} & 0\\ a_{21} & a_{22} & 0\\ 0 & 0 & 1\end{bmatrix}T(x_0,y_0)$

4. **三维坐标系之间的变换：**

1) 坐标系原点重合，从世界坐标系变换到辅助坐标系：

$$T_R=\begin{bmatrix}a_{11} & a_{21} & a_{31} & 0\\ a_{12} & a_{22} & a_{32} & 0\\ a_{13} & a_{23} & a_{33} & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

2) 坐标系原点重合，从辅助坐标系变换到世界坐标系：

$${T_R}^{-1}=\begin{bmatrix}a_{11} & a_{12} & a_{13} & 0\\ a_{21} & a_{22} & a_{23} & 0\\ a_{31} & a_{32} & a_{33} & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

3) 坐标系原点不重合，从世界坐标系变换到辅助坐标系：

$$T_R=T(-x_0,-y_0,-z_0)\begin{bmatrix}a_{11} & a_{21} & a_{31} & 0\\ a_{12} & a_{22} & a_{32} & 0\\ a_{13} & a_{23} & a_{33} & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

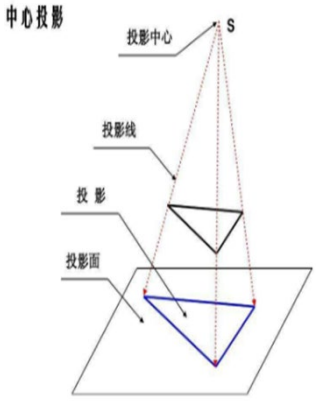
4) 坐标系原点不重合，从辅助坐标系变换到世界坐标系：

$${T_R}^{-1}=\begin{bmatrix}a_{11} & a_{12} & a_{13} & 0\\ a_{21} & a_{22} & a_{23} & 0\\ a_{31} & a_{32} & a_{33} & 0\\ 0 & 0 & 0 & 1\end{bmatrix}T(x_0,y_0,z_0)$$

第六章 · 投影变换

➢ 投影

1. 投影的实质是将一个 n 维的空间点变成一个小于 n 维的空间点，投影与三个因素有关：空间物体的位置，观察点（焦点、镜头、眼睛）的位置或投影方向，投影面的位置。
2. **三维形体的表示方法**：线框模型、表面模型和实体模型。
   1) **线框模型**：以一组或几组轮廓线来表示形体。它是实体的高度抽象，容易表现实体的拓扑结构；
   2) **表面模型**：由形体表面来表示形体。形体的表面有法向，有纹理；面与面之间存在遮挡关系。是三维城市模型中常用的方式；
   3) **实体模型**：用形体各种组成物体实际占据的空间位置，以及材料、质量、重心等物理属性来表示形体。在医学研究中常用。
3. **平面几何投影**：
   1) **相关概念**：
      A. 投影中心：观察点；
      B. 物点：空间物体上的任意一点；
      C. 投影线：投影中心与物点的空间连接直线；
      D. 投影平面：投影成像的平面；
      E. 像点：投影线与投影平面的交点就是物点的像。



   2) **平面几何投影分类：透视投影和平行投影**：
      A. 投影中心到投影平面的距离有限时，就是透视投影，眼睛、摄像机观察世界就是透视投影方式；
      B. 当投影中心距离投影平面无穷远时，所有的投影线彼此平行，为平行投影。太阳光照射形成的投影就是平行投影
4. **透视投影**：
   1) 透视投影属于中心投影，所有的投影线都是由焦点（观察点）发出；
   2) 透视投影图简称为透视图或透视；
   3) 透视投影的特点是近大远小，不平行于投影面的平行线，经投影后不再平行
   4) 透视投影变换不是仿射变换，不是线性的。
   5) 不平行于投影面的平行线，经过透视投影之后交会于一点，称为灭点。
      A. 灭点交会于无穷远点处。

B. 平行于投影平面的平行线，经过透视投影以后依然平行，它们没有灭点。

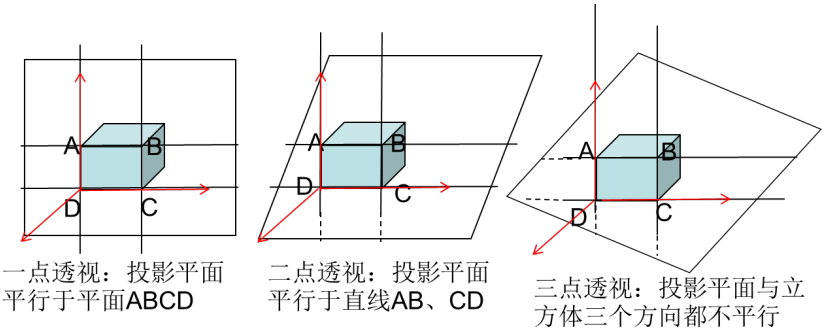
C. 主灭点：落在坐标轴上的灭点称为主灭点。

D. 主灭点是与坐标轴平行的线相交的灭点。

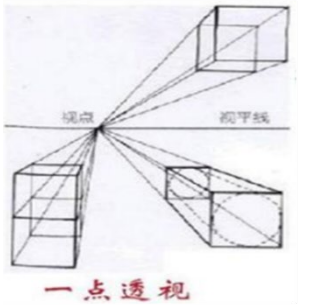
6) 透视投影按照主灭点数分为一点透视，二点透视，三点透视。

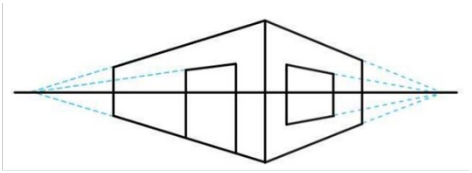
☆一点透视：有一个坐标轴与投影平面不平行；二点透视：有两个；三点有三个

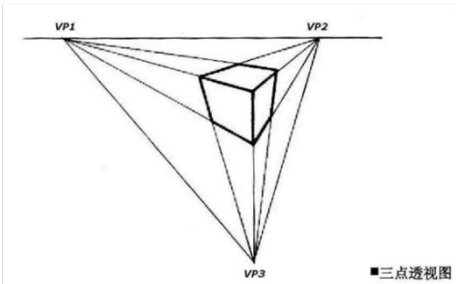
摄像时应根据要表现的内容选择适当的角度（实际是选择投影面使其与坐标轴平行或不平行），形成不同效果的透视图。



一点透视：投影平面平行于平面ABCD　二点透视：投影平面平行于直线AB、CD　三点透视：投影平面与立方体三个方向都不平行







■三点透视图

1) **构造一点透视的一般步骤如下**：将物体平移到适当位置，进行透视变换，最后向 xoy 平面做正投影，即可得到一点透视图。

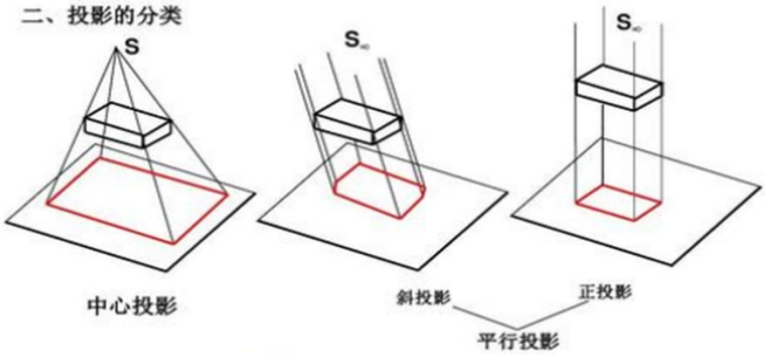
2) **构造二点透视的一般步骤如下**：将物体平移到适当位置，将物体绕 y 轴旋转 θ 角，进行透视变换，最后向 xoy 平面做正投影，即可得到二点透视图。

3) **构造三点透视的一般步骤如下**：将物体平移到适当位置，将物体绕 y 轴旋转 θ 角，再绕 x 轴旋转 α 角，进行透视变换，最后向 xoy 平面做正投影，即可得到三点透视图。

5. **平行投影：**如果把中心投影法的投射中心移至无穷远处，则各投射线成为相互平行的直线，这种投射线都互相平行的投影法称为平行投影法。

1) 正投影：投影线垂直于投影面。常见的正投影有正射影像和三视图

2) 斜投影：投影线不垂直于投影面



➢ 投影变换

1. 投影变换表现了从物点到像点的数学关系。

2. 在坐标系$o'x'y'z'$中讨论投影。假设投影平面为$z'=0$，投影中心为$C'(xc',yc',zc')$，物点为$Q'(x',y',z')$，在投影平面$z'=0$上的像点为$P'(xp',yp',0)$。显然，C'，Q'，P'三点在同一空间直线上。

3. **$z=0$平面的透视投影变换公式推导**:

过空间两点$(x1,y1,z1)$，$(x2,y2,z2)$的空间直线方程为：$\frac{x-x_1}{x_2-x_1}=\frac{y-y_1}{y_2-y_1}=\frac{z-z_1}{z_2-z_1}$

由于$P'(xp',yp',0)$为空间直线$C'Q'$上的点，所以满足方程：$\frac{x_p{}'-x_c{}'}{x'-x_c{}'}=\frac{y_p{}'-y_c{}'}{y'-y_c{}'}=\frac{0-z_c{}'}{z'-z_c{}'}$

整理得透视投影计算公式:

$$x_p{}'=x_c{}'+(x'-x_c{}')\frac{z_c{}'}{z_c{}'-z'}\ ;\ y_p{}'=y_c{}'+(y'-y_c{}')\frac{z_c{}'}{z_c{}'-z'}\quad(6-1)$$

4. **$z=0$平面的平行投影变换公式推导**:

过空间物点$Q'(x',y',z')$，沿投影方向(xd',yd',zd')作一空间直线，直线与投影平面$z'=0$的交点就是投影点$P'(xp',yp',0)$。

投影线的空间直线方程为：$\frac{x-x'}{x_d}=\frac{y-y'}{y_d}=\frac{z-z'}{z_d}$

将$P'(xp',yp',0)$代入，整理得平行投影计算公式

$$x_p{}'=x'-\frac{x_d}{z_d}z'\ ;\ y_p{}'=y'-\frac{y_d}{z_d}z'\quad(6-2)$$

用矩阵表示：$[x_p' \quad y_p' \quad 1] = [x' \quad y' \quad z' \quad 1]\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -\frac{x_d'}{z_d'} & -\frac{y_d'}{z_d'} & 0 \\ 0 & 0 & 1 \end{bmatrix}$

5. **任意投影平面的投影：**

1) 在坐标系$oxyz$中，当投影平面为任意平面时，可以在投影平面上作辅助坐标系$o'x'y'z'$使投影平面为$z'=0$平面。在进行投影前，将空间物体从坐标系$oxyz$中变换到坐标系$o'x'y'z'$中，然后就可以按照前述的投影平面为$z'=0$平面的方式进行投影变换。

2) 坐标系$o'x'y'z'$的确定不是任意的。一般为了后续的图形裁剪与显示的方便，要求x'、y'轴分别平行于裁剪窗口的两条边，$o'x'y'z'$坐标原点位于裁剪窗口的左下角。

✧ 即不需要再从辅助坐标系变换回世界坐标系。

3) **从$oxyz$到$o'x'y'z'$的变换关系**：

$$[x' \quad y' \quad z'] = [x-x_0 \quad y-y_0 \quad z-z_0]\begin{bmatrix} a_{11} & a_{21} & a_{31} \\ a_{12} & a_{22} & a_{32} \\ a_{13} & a_{23} & a_{33} \end{bmatrix}$$

用齐次坐标表示：

$$[x' \quad y' \quad z' \quad 1] = [x \quad y \quad z \quad 1]\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -x_0 & -y_0 & -z_0 & 1 \end{bmatrix}\begin{bmatrix} a_{11} & a_{21} & a_{31} & 0 \\ a_{12} & a_{22} & a_{32} & 0 \\ a_{13} & a_{23} & a_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6-3)$$

注：a13 是 x'轴单位向量在 Z 轴上的投影；

a23 是 y'轴单位向量在 Z 轴上的投影；

a31 是 z'轴单位向量在 X 轴上的投影；

a32 是 z'轴单位向量在 Y 轴上的投影；

a33 是 z'轴单位向量在 Z 轴上的投影；

6. **任意投影平面的透视投影变换**：

透视投影计算公式去掉分母变换为线性关系：

$$x_q' = (z_c' - z')x_p' = x_c'(z_c' - z') + (x' - x_c')z_c'$$

$$y_q' = (z_c' - z')y_p' = y_c'(z_c' - z') + (y' - y_c')z_c'$$

用矩阵表示为：$[x_q' \quad y_q' \quad q] = [x' \quad y' \quad z' \quad 1]\begin{bmatrix} z_c' & 0 & 0 \\ 0 & z_c' & 0 \\ -x_c' & -y_c' & -1 \\ 0 & 0 & z_c' \end{bmatrix}$ (6-4)

与任意投影平面的投影公式矩阵做合并得：

$[x_q' \quad y_q' \quad q] =$

$$[x \quad y \quad z \quad 1]\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -x_0 & -y_0 & -z_0 & 1 \end{bmatrix}\begin{bmatrix} a_{11} & a_{21} & a_{31} & 0 \\ a_{12} & a_{22} & a_{32} & 0 \\ a_{13} & a_{23} & a_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} z_c' & 0 & 0 \\ 0 & z_c' & 0 \\ -x_c' & -y_c' & -1 \\ 0 & 0 & z_c' \end{bmatrix} \quad (6-5)$$

(xq', yq')不是投影点，作如下计算得到投影点：

$$x_p' = x_q'/q \quad ; \quad y_p' = y_q'/q \quad (6-6)$$

✧ 注意：观测点坐标(xc', yc', zc')是在辅助坐标系$o'x'y'z'$中的坐标。实际问题中给出的是$oxyz$中的坐标(xc, yc, zc)，应该用式(6-3)将其转变成为辅助坐标系的坐标

7. **任意投影平面的平行投影变换**：

(6-2)与(6-3)合并矩阵即可：

$$[x_p' \quad y_p' \quad 1]$$

$$= [x \quad y \quad z \quad 1]\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -x_0 & -y_0 & -z_0 & 1 \end{bmatrix}\begin{bmatrix} a_{11} & a_{21} & a_{31} & 0 \\ a_{12} & a_{22} & a_{32} & 0 \\ a_{13} & a_{23} & a_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -\frac{x_d'}{z_d'} & -\frac{y_d'}{z_d'} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= [x \quad y \quad z \quad 1]\begin{bmatrix} b_{11} & b_{21} & b_{31} \\ b_{12} & b_{22} & b_{32} \\ b_{13} & b_{23} & b_{33} \\ b_{14} & b_{24} & b_{34} \end{bmatrix}$$

➢ 观察坐标系下的一点透视

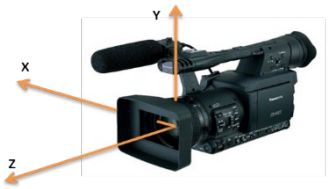
1. 三维场景漫游，视点（摄像机，照相机）在场景中移动，场景在二维感光面成二维图像（投影），其随视点空间位置和视点方向的变化而发生变换。场景漫游的视点是摄像机，摄像机是有限制的，就是投影平面是固定的，不是随意变化的，是与摄像机光轴垂直的。
2. **观察坐标系**：是一个原点在投影中心（视点）的左手直角坐标系。
3. **为什么要采用观察坐标系**：便于用户选择好的视点；适应观察时要求物体不动而视点动的应用需求；简化和加速投影变换。
4. **如何建立观察坐标系**

1) 坐标原点----聚焦参考点在底片（投影平面）上的投影，称为观察参考点 VRP（View Reference Point）

2) z 轴----照相机镜头方向（投影平面的法向，VPN）

3) y 轴----照相机向上的方向（观察正向）

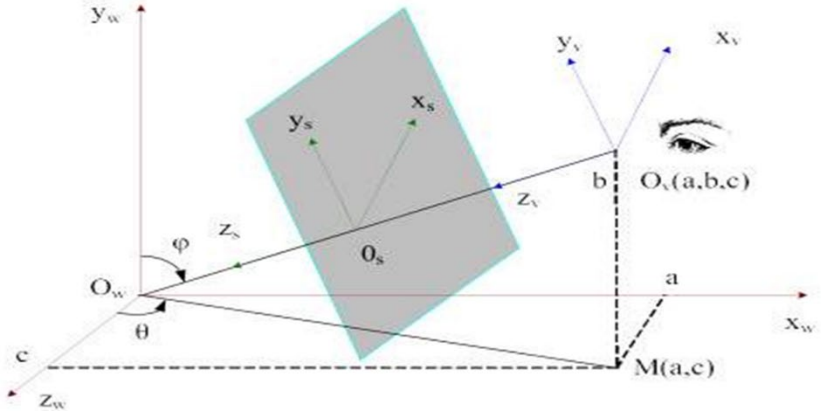
4) x 轴----与 y、z 轴垂直的方向



5. **坐标系之间的关系**

$O_WX_WY_WZ_W$ 世界坐标系，$O_VX_VY_VZ_V$ 观察坐标系，$O_SX_SY_S$ 屏幕坐标系。

屏幕就是一点透视的投影平面。



6. **观察坐标系下透视投影公式**:

过空间两点$C(0,0,0)$、$Q(xq,yq,zq)$的空间直线方程为:

$$\frac{x}{x_q}=\frac{y}{y_q}=\frac{z}{z_q}$$

投影点$P(xp,yp,zp)$满足直线方程，其中$zp = d$

$$x_p = d\times\frac{x_q}{z_q};\ y_p = d\times\frac{y_q}{z_q}\quad(6-8)$$

7. **如何从用户坐标系到观察坐标系**：必须首先将模型从用户坐标系变换的观察坐标系（6-3），再应用式（6-8）计算投影坐标。

8. **场景漫游的原理**:

1) 摄像机在三维场景中不断连续变换空间位置和指向方向;

2) 在这个动态过程的某个时刻，摄像机的空间位置(x0,y0,z0)和姿态（3 个观察坐标轴 9 个方位参数）都是确定的。在这个时刻，利用这 12 个参数，将三维场景投影到观察坐标系投影面上;

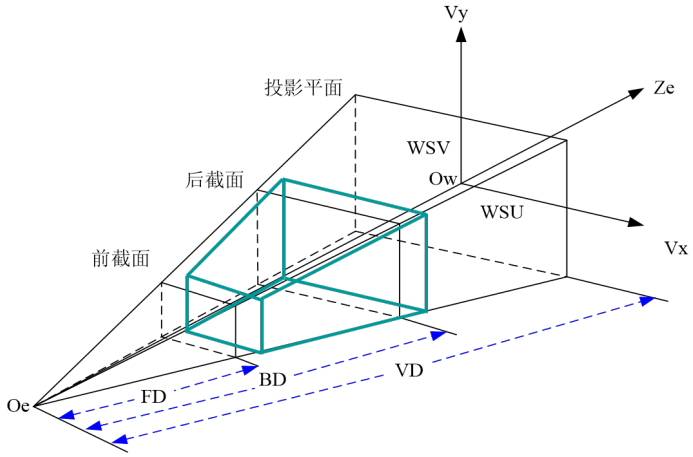
3) 整个漫游过程，按照规定的时间间隔，有很多这样的时刻，需要依次进行投影。规定的时间间隔足够小，就能形成连续的视频效果;

4) 投影计算简单，但模型变换计算量大，且随着照相机的漫游，12 个参数发生变化，从用户坐标系到观察坐标系的变换矩阵需要重新计算。模型变换关系不断变化，需要不断计算。

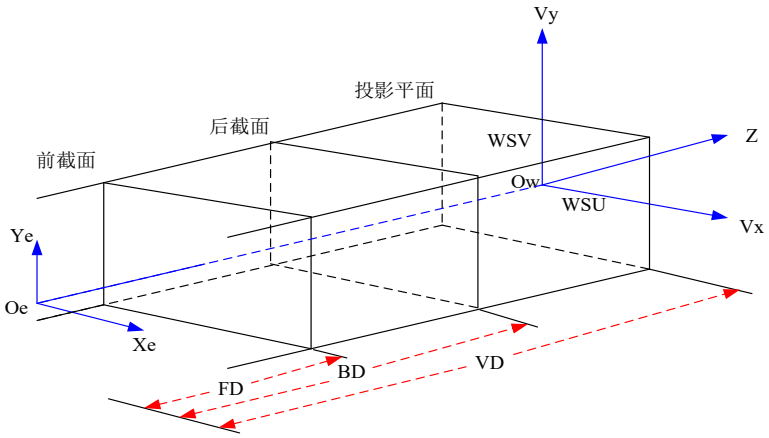
➢ 投影空间

1. **投影空间：**是一个有限的三维空间，在该空间中的任何物体经过投影后，都将落在二维窗口中，只有投影空间的模型才有可能被观察到。投影空间的作用是将整个场景的所有模型分成投影空间内模型和投影空间外模型两个部分，这就需要用投影空间对整个场景进行三维裁剪。相对于二维的窗口概念，三维的投影窗口称为投影空间。
2. **根据投影方式，有两种投影空间**：

   1) 透视投影空间：四棱台体



   2) 平行投影空间：四棱柱体



3. **规格化的裁剪空间和图像空间**：

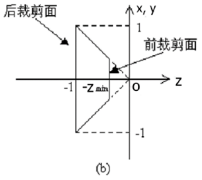
   任意的透视投影，其投影空间为斜四棱台

任意的平行投影，其投影空间为斜四棱柱

它们是六面体，由六个平面组成

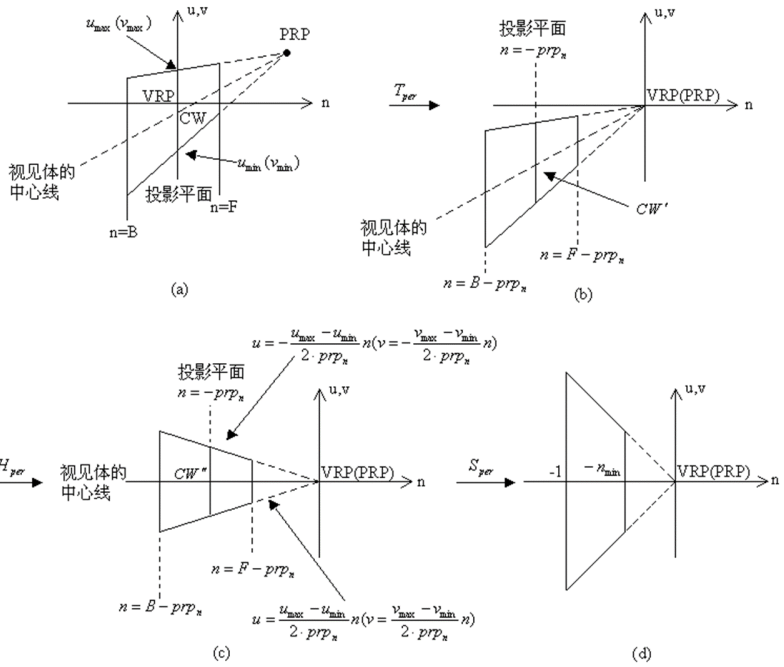
斜四棱台、斜四棱柱面平面方程表示不规范，求交、裁剪的效率不高，将斜四棱台规格化为正四棱台，斜四棱柱规格化为正四棱柱十分必要。

4. **规范视见体——正四棱台**

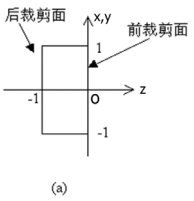


由六个平面组成：$\begin{cases} x=z, x=-z \\ y=z, y=-z \\ z=-z_{min}, z=-1 \end{cases}$

规格化：

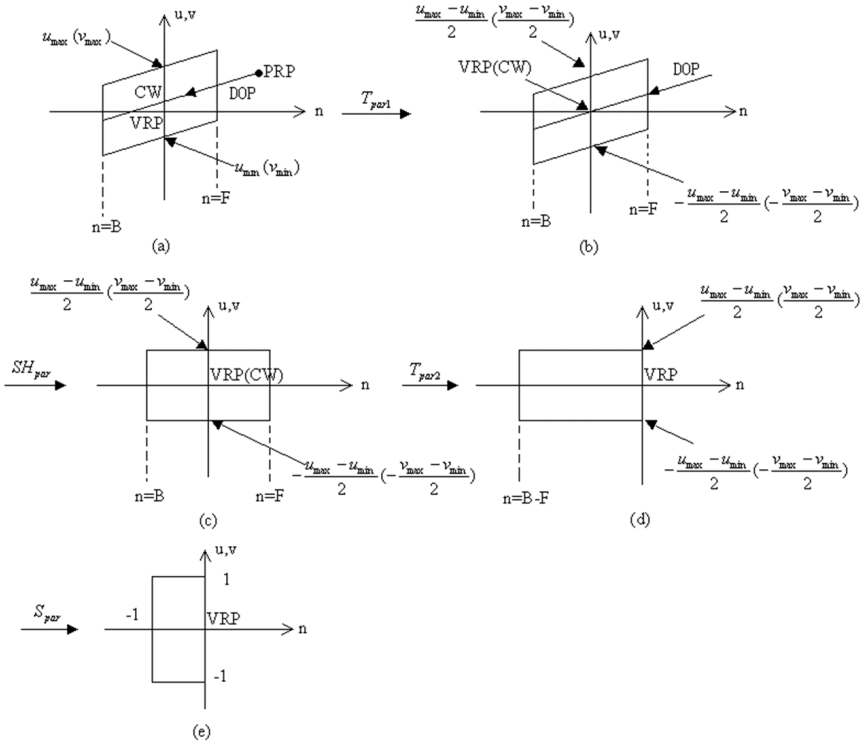


5. **规范视见体——半立方体**



由六个平面组成：$\begin{cases} x=1, x=-1 \\ y=1, y=-1 \\ z=0, z=-1 \end{cases}$

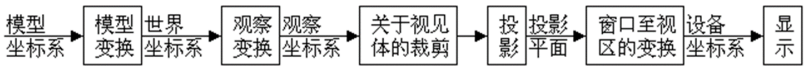
规格化：



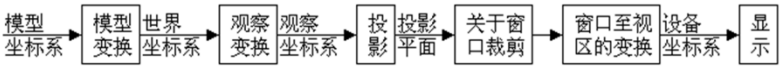
➢ 三维图形的显示

1. **两种裁剪方式：三维裁剪和二维裁剪**

   1) 采用三维裁剪的三维图形显示流程图



   2) 采用二维裁剪的三维图形显示流程图



2. 依据何时裁剪来看：

   1) **投影之前裁剪----三维裁剪**：

   裁剪结果经过投影后，都在窗口范围内，都是可见的。

   优点 ： 只对可见的物体进行投影变换，大幅度减少投影计算量；

   缺点 ： 三维裁剪相对复杂。

   2) **投影之后裁剪----二维裁剪**：

   先将模型中所有物体都投影到投影平面，然后用二维投影窗口进行裁剪。

优点：二维裁剪相对容易；

缺点：需要对所有的物体进行投影变换，投影计算量大。

3. **三维线段裁剪**

立方体窗口六个面的方程分别是：$x=1, x=-1, y=1, y=-1, z=0, z=1$

三维线段参数方程：$x=x0+(x1-x0)t;\ y=y0+(y1-y0)t;\ z=z0+(z1-z0)t;$

根据 x0 与 $x1$，$y0$ 与 $y1$，$z0$ 与 $z1$的大小关系分别确定 3 个始面，3 个终面

参数方程与 3 个始面分别联立，求出直线与面的交点参数$t1', t2', t3'$

参数方程与 3 个终面分别联立，求出直线与面的交点参数$t1'', t2'', t3''$

$$t0=max(0,t1',t2',t3'),\qquad t1=min(1,t1'',t2'',t3'')$$

如果$t0<t1$，计算$t0$、$t1$对应的点，得到的线段为裁剪结果

第七章 · 隐藏面的消除

➢ 基本概念

1. **三维物体的三种模型：线框模型，表面模型，实体模型**。

线框模型：涉及隐藏线的消隐。

表面模型：涉及隐藏面的消隐。

实体模型（表面模型+内部内容）：只需考虑实体表面的问题，与表面模型相似

2. 在点、线、面、体元素中，能产生遮挡作用的面、体，体是由面组成的，基本的遮挡单元是面。在消隐计算中，物体分解成组成物体的各个表面，这些表面用空间平面多边形（简称为面）表示，并以每个空间多边形为独立单元进行消隐计算。

3. **各种消隐算法的三个统一前提**：

1) 消隐单元为面（有限空间平面），考虑有限平面的遮挡作用；

2) 投影方式：面向 Z=0 平面的正射投影，投影方向为 Z 轴的反方向；

3) 在数据库中众多的三维模型中，只考虑能落入窗口的物体的遮挡作用，这是为了减少计算量。在消隐算法开始前，用包围盒从众多的三维模型中，选出落入窗口的三维模型

4. 沿着视线方向，只有前面的面才有可能遮挡后面的面。前面的平面不一定遮挡或完全

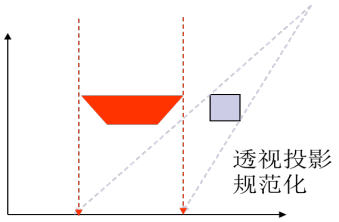
遮挡后面的平面，这样就要将不被遮挡的部分显示出来，判断两个物体是否相互遮挡，可以在物体投影到窗口平面后，用物体投影的边界盒是否相交来判断。

5. 前向面：外法线方向与视线方向成钝角；后向面：外法线方向与视线方向成锐角
只需要考虑前向面的遮挡作用；对于后向面，不需要考虑【成直角也不需要考虑】

6. **投影规范化：**为了简化消隐处理，在消隐前将各种投影转化为正射投影的操作。
将点$p(x,y,z)$投影的结果(xp,yp)，看作是点$p'(xp,yp,z)$。即将点 p 的 z 直接赋予点p'，就完成了投影的规范化。

➢ 线框图消隐（隐藏线消隐）：

线框图形是用棱线来表示物体，棱线是两个面的交线消隐计算必须一个一个多边形依次进行，对于一个多边形的消隐必须一条边一条边地进行。基本方法是依次将每根棱线与多边形相比较，确定棱线被多边形遮挡的部分。



隐藏线就是被一个或几个多边形遮挡住的棱线部分；

消除隐藏线就是找出被遮挡的棱线部分，只显示未被遮挡的棱线部分。

减小计算量：

1) 用边界盒、包围盒比较棱线与每个空间多边形的位置关系，排除绝对不遮挡棱线的空间多边形；二维边界盒不相交，不可能形成遮挡关系（因为是垂直投影，只需要考虑物体的 x,y 坐标）：用二维边界盒判断：$xmin1 > xmax2$ 或$xmin2 > xmax1$ 或$ymin1 > ymax2$ 或$ymin2 > ymax1$

2) 位于一条棱线后面的物体不可能遮挡该棱线，用三维包围盒判断：$Zmin线 > Zmax体$【前提是投影平面为 Z=0，投影方式为垂直平行投影】

3) 后向面（自隐藏面）可以全部去掉

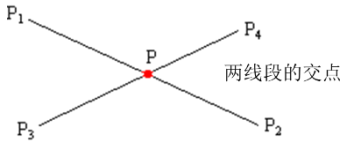
求外法线：

设物体位于右手坐标系中，多边形顶点按逆时针排列，设法矢量$\{A,B,C\}$，则：

$$A=\sum_{i=1}^{n}(yi-yj)\times(zi-zj)\ ;\ B=\sum_{i=1}^{n}(zi-zj)\times(xi-xj)\ ;\ C=\sum_{i=1}^{n}(xi-xj)\times(yi-yj)$$

式中：n 为顶点号，若$i\neq n$，则$j=i+1$；否则$i=n$，$j=1$。 即i，j表示相邻的两个顶点

求两线段的交点：



P 点的参数方程：

$$\begin{cases}x=x1+(x2-x1)u\\y=y1+(y2-y1)u\end{cases},0\le u\le 1\quad(8\text{-}1)\ ;\ \begin{cases}x=x3+(x4-x3)v\\y=y3+(y4-y3)v\end{cases},0\le v\le 1\quad(8\text{-}2)$$

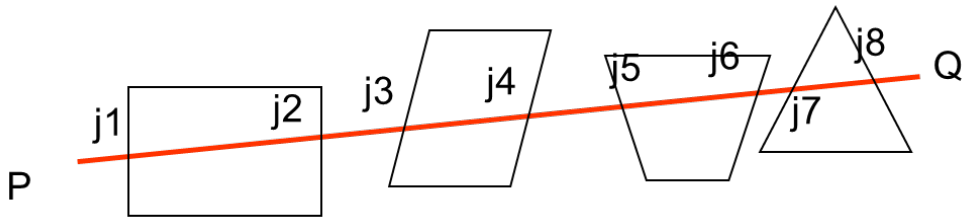
两直线的交点应该满足：$\begin{cases}x1+(x2-x1)u=x3+(x4-x3)v\\y1+(y2-y1)u=y3+(y4-y3)v\end{cases}$

$$u=\frac{\begin{vmatrix}x_3-x_1 & x_3-x_4\\ y_3-y_1 & y_3-y_4\end{vmatrix}}{\begin{vmatrix}x_2-x_1 & x_3-x_4\\ y_2-y_1 & y_3-y_4\end{vmatrix}}\qquad(8\text{-}3)$$

$$v=\frac{\begin{vmatrix}x_2-x_1 & x_3-x_1\\ y_2-y_1 & y_3-y_1\end{vmatrix}}{\begin{vmatrix}x_2-x_1 & x_3-x_4\\ y_2-y_1 & y_3-y_4\end{vmatrix}}\qquad(8\text{-}4)$$

最终，若参数值同时满足$0\le u\le 1$，$0\le v\le 1$，则两线段有交点，代入公式（8-1）或者（8-2）即可求得交点坐标（x，y）。（注意先判断分母不为零）

可见部分的确定：



1) 将所有交点按照参数从小到大排列；对应参数为：$l1,l2,l3,\dots,lk$。

2) 设初始不可见阶参数$ivord=0$

3) 线段起点 P 处的不可见阶$ivord$的确定方法：分别考虑线段 PQ 与所有多边形的第一个交点，如果为“出点”，$ivord+1$；如果为“进点”，$ivord$不变。

4) 如果 P 处$ivord=0$，则 PJ1 段可见，否则，不可见。

5) J1 的不可见阶$ivord$，若 J1 为“出点”，$ivord-1$；J1 为“进点”，$ivord+1$。

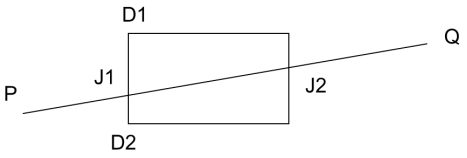
6) 若$ivord$不为 0，则 J1J2 段不可见，否则可见。

7) 按照 5、6 依次处理余下的各个交点，直到每一段可见性判断完毕。

出点与进点：

出点：线段出多边形处的点。

进点：线段进多边形处的点。



$\vec{R} = \overrightarrow{D_i D_{i+1}} \times \overrightarrow{PQ}$ ；（D_i各点按逆时针排序，右手螺旋法则，拇指指向外法线方向）

向量 R 在 Z 轴上的投影为负，进点（J1）；向量 R 在 Z 轴上的投影为正，出点（J2）

➢ 曲面隐藏线消除：

方法：从前向后依次处理每一条曲线，每一条曲线露出（高出前面所有曲线）的部分被画出来。

DEM 与影像数据类似，由排列整齐的点阵数据阵列组成。数据所在的行列表明位置信息，用(x, y)表示，数据本身 z 表示高程值。由这些空间点依次连接所得到的折线可以表示曲面上的一条曲线。

表示曲面地形的一种方法是：将每一行、每一列的空间点分别依次连接起来，得到的若干条空间曲面上的曲线，并投影到投影平面上，构成 DEM 所表示的空间曲面。

做 DEM 消隐的绘制方法：

1) 绘制空间点连线的方法同前一种方法，只是为了消隐，绘制次序有规定；
2) 绘制次序是按照距离视点的远近，从近到远依次绘制，以实现前图遮挡后图的正确遮挡效果；
3) 后图只有高出前图，才不会被前图遮挡；
4) 为了只绘制未被遮挡的曲线部分，设置一个高度记录器，依次记录已绘制的最高点值y_p值【每一行、列都有一个高度记录器】；
5) 只有超出高度记录器的曲线部分被绘制，并且其高度值y_p值代替原有的高度值记录进记录器；
6) 考虑到前行（列）可以遮挡后列（行），依据离视点的远近程度，按照从前到后（行）、从左到右（列）的顺序，交替绘制行、列曲线；
7) 其中超出高度记录器范围的曲线被绘制，并且其高度记录进高度记录器，未超出

的部分不绘制；

8) 高度记录器为一个一维数组，数组单元数量与显示器视区宽度相同，即每个单元对应于一个像素，也就是每个单元对应于 X 轴整数部分；

9) 相邻两个空间点投影坐标(x_{pi}, y_{pi})、(x_{pi+1}, y_{pi+1})中的y_p值必须线性内插到 x 整数坐标上。y_p不是整数值

10) 备注：为了不遮挡“漏斗”型曲面，设置两个高度记录器，分别记录最大最小值

➢ 画家算法（面消隐）：

1. **算法的基本思想是**:

1) 先将场景中的物体（空间面）按其距观察点的远近进行排序，排序结果存在一张深度优先表中。

2) 距观察点远者称其优先级低，放在表头；
距观察点近者称其优先级高，放在表尾。

3) 按照从表头到表尾的顺序逐个绘制物体，近的物体后画，覆盖了先画的远的物体，产生了正确的遮挡关系

2. **运用画家算法前，采用以下方法，减少计算量：**

1) 用三维包围盒确定能够投影到窗口的空间范围，除去包围盒以外的物体。

2) 每个物体去除后向面，只剩下前向面，每个前向面用多边形顶点表示。

3. 画家算法的关键和难点在于如何对多边形按远近进行排序。

每个多边形顶点中，最大的纵坐标 z 值用z_{max}表示，最小的纵坐标 n 值用z_{min}表示；根据每个多边形的z_{min}对它们进行预排序，并放入优先级表中；这样大部分多边形都是按照远近关系得到排序，但有少量特例例外。

为了排除例外，需要逐个进行两两比较，判断多边形之间的远近关系。设 P 是在优先级表中排列第一的多边形，Q 为任意一个其他多边形。 Q 与 P 进行比较，若 Q 在 P 的后方，P 比 Q 距离视线更近，需要交换 Q 和 P 的位置。

4. 前后方判别方法：

1) P 面投影边界盒$(x1, y1, x2, y2)$，Q 面投影边界盒$(x3, y3, x4, y4)$；
2) 如果两个边界盒没有重叠，说明 P、Q 面相互之间没有影响，因此没有必要比较前后关系；退出
3) 两个边界盒的重叠部分表示为（$x_{min}, y_{min}, x_{max}, y_{max}$），选一个点$((x_{min}+x_{max})/2, (y_{min}+y_{max})/2)$，该点在 P、Q 面上有对应点；
4) 将该点分别代入 P、Q 面方程$z1 = p(x, y)$，得到$z2 = Q(x, y)$；
5) $z2 > z1$，Q 面在 P 面前方；$z2 < z1$，Q 面在 P 面后方

➢ Z 缓冲器算法：

Z 缓冲器是一组存储单元，其单元个数与屏幕上的像素个数相同，与帧缓冲器（显存）单元个数相同。它们三者存在一一对应的关系。

Z 缓冲器算法为每一个要填入的像素，不光准备颜色值，还准备一个深度值。

屏幕上像素的显示颜色由对应的帧缓冲器单元中的数值决定，该像素对应的 z 值存储在 Z 缓冲器对应单元中。

有多个空间物体对应同一个显示像素。一个颜色值是否能填入显示缓冲器像素，取决于该颜色值所对应的深度值是否大于当前缓冲器中的深度值

深度值的获取：

1) 由多边形的任意三个顶点，可以确定平面方程；
2) 将多边形投影到投影平面，并着色；
3) 在着色的过程中可以确定多边形的每一个内点坐标；
4) 由内点坐标和平面方程可以确定内点对应的纵坐标（深度值）。

Z 缓冲器算法的缺点是需要太多的缓冲存储空间，要准备一个与屏幕像素对应的深度 Z 值存储空间。

➢ 扫描线 Z 缓冲器算法：

为了克服 Z 缓冲器算法需要太多的缓冲存储空间的缺点，可以把整个显示平面分成若干个区域，一区一区来显示，这样 Z 缓冲器的单元数只要等于一个区域的像素个数。

如果把区域变成显示屏幕的一行，就得到扫描线 Z 缓冲器算法。

缓冲区由区域变成水平扫描线，问题发生质的变化：

使用一个多边形表(PT)对所有的多边形排序、分类；

使用一个活化多边形表(APT)列出与当前扫描线相交的多边形；

为每个多边形建立一个边表(ET)；

使用一个活化边表(AEPT)列出与当前扫描线相交的边对。

➢ 区域子分算法

1. **显示窗口与投影多边形的四种关系**：
   1) 多边形包围了窗口；
   2) 多边形与窗口相交；
   3) 窗口包围了多边形；
   4) 窗口与多边形分离。
2. **所有的多边形与窗口的关系确定以后，以下三种情况下就可以确定颜色了**：
   1) 所有多边形与窗口分离，窗口填上背景色；
   2) 只有一个多边形与窗口相交或包含在窗口内，先对窗口填上背景色，再运用多边形扫描线方法对多边形内部填上该多边形的颜色；
   3) 离观测者最近的多边形包围了窗口，对窗口填上该多边形的颜色。

   只有满足上述三种情况，我们才能够处理。除此之外，我们无能为力。
3. 区域子分算法的处理方法是，将窗口一分为四，再判断每个子窗口与多边形的关系。这样，一些多边形与子窗口的关系成为上述三种关系中的一种。如果子窗口与多边形的关系还不能满足三种情况之一，再将该窗口一分为四。如此重复，直到所有的子窗口与多边形的关系能满足三种情况之一为止。

第八章·明暗效应与颜色模型

➢ 明暗效应

1. 定义：光照射到物体表面所产生的反射或透射现象的模拟。
2. 从两大类因素入手：

1) 光强度越大，物体越亮。（明暗效应就是计算反射光或折射光的光强，且必须将所有类型的反射光都考虑到）；

2) 物体自身因素，物体对入射光都有一定程度的吸收作用，而且是选择性地吸收。不同的物体吸收效果差异很大。物体表面的光滑程度对光的反射也有很大的影响

3. **泛光：**入射光在环境中经过多次反射的结果。在空间中近似均匀分布，即在任何位置、任何方向上强度一样，是一个常数I_a。

4. **光照明方程：** $I = K_a \times I_a$

I_a：入射的泛光光强，与环境的明暗度有关

K_a：漫反射系数，在分布均匀的环境光照射下，不同物体表面所呈现的亮度未必相同，因为它们的漫反射系数不同。

5. **漫反射光：**粗糙、无光泽物体（如粉笔）表面对光的反射

1) **点光源的照射**：照在物体的不同部分其亮度也不同，亮度的大小依赖于物体的朝向及它与点光源之间的距离；

2) **特点**：空间分布是均匀的。

3) **兰伯特（Lambert）余弦定律**：反射光强与入射光的入射角的余弦成正比。

4) **光照明方程：** $I = K_d I_l \cos\theta \qquad \theta \in \left[0, \frac{\pi}{2}\right]$

I_l点光源的亮度　　K_d漫反射系数　　θ入射角

漫反射光的强度只与入射角有关

6. **镜面反射光：**光滑物体（如金属或塑料）表面对光的反射

1) 高光：入射光在光滑物体表面形成的特别亮的区域

2) 光照明方程：$I = I_l K_s \cos^n \alpha$

K_s：物体镜面反射系数　　α：视线与反射方向的夹角

n：镜面反射指数或会聚指数，n 越大，物体越光滑。

➢ 均匀着色与光滑着色

1. **均匀着色与光滑着色**

1) 均匀着色

所适用场景：①光源在无穷远处；②视点在无穷远处；③多边形是物体表面的精确表示（表面平整没有凹凸）；

2) 光滑着色，亦称插值着色

2. **曲面均匀着色：**将曲面体用多边体近似表示，再进行均匀着色

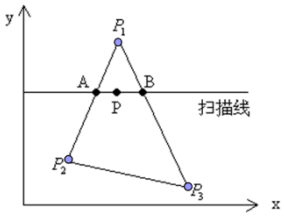
存在两个问题：

1) 把曲面体离散成很细的小面片：存储容量与处理时间耗费很大

2) 把曲面体离散成很粗的面片：产生马赫带效应

解决：边界处光滑着色

3. **光滑着色(插值着色)——Gouraud 方法**



通过对多边形顶点颜色进行线性插值来获得其内部各点光强度。

对多边体中的每一个多边形，其着色步骤如下：

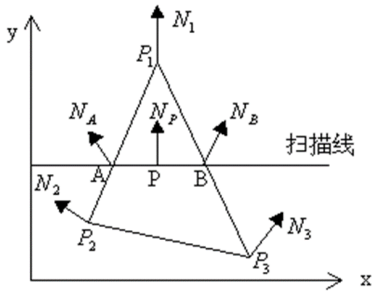
1) 计算多边形顶点的单位法矢量；

2) 利用光照明方程计算顶点的光强度；

3) 在扫描线消隐算法中，对多边形顶点的光强值进行双线性插值，获得多边形内部（位于多边形内的扫描线上）各点的光强度。

4. **光滑着色(插值着色)——Phong 方法**

通过对多边形顶点的法矢量进行线性插值来获得其内部各点的法矢量，又称法向插值着色方法。



步骤如下：

1) 计算多边形顶点的单位法矢量；

2) 在扫描线消隐算法中，对多边形顶点的法矢量进行双线性插值，获得多边形内部（位于多边形内的扫描线上）各点的法矢量；

3) 利用光照明方程计算各点的光强度。

两方法的比较：

1) Phong 着色方法计算量远大于 Gouraud 着色方法。

2) Phong 着色方法绘制的图形比 Gouraud 方法更真实。

5. 简单透明不考虑折射现象：$I_\lambda = (1 - K_{t_1})I_{\lambda 1} + K_{t_1} I_{\lambda 2}$

➢ 颜色模型

1. **颜色：**物体将光源投射光反射到人的眼睛中的光刺激作用于人的视觉器官而产生的主观感觉。
2. **颜色特性**
   1) **从视觉的角度出发，**颜色有如下三个特性：色彩（Hue）,饱和度（Saturation）和亮度（Lightness）。
      a) 色彩：是一种颜色区别于其他颜色的因素：红、绿、蓝、紫等；
      b) 饱和度：是指颜色的纯度，鲜红色的饱和度高，而粉红色的饱和度低；
      c) 亮度：就是光的强度，是光给人的刺激的强度。
   2) **从物理学的角度出发**，颜色有如下三个特性：主波长，纯度和辉度。
      a) 主波长：所见颜色光的波长，对应于色彩；
      b) 纯度：是指颜色的纯度，对应于饱和度；
      c) 辉度：就是颜色的亮度。
3. CIE 色度图
   1) 三原色（三基色）

      三维颜色空间的一组基，三原色的条件：

      a) 用适当比例的这三种颜色，可以获得白色
      b) 用这三种颜色中的任意两种的组合都不能得到第三种颜色
      c) 用适当比例的这三种颜色，可生成其他颜色

      三基色的例子：红、绿、蓝三基色

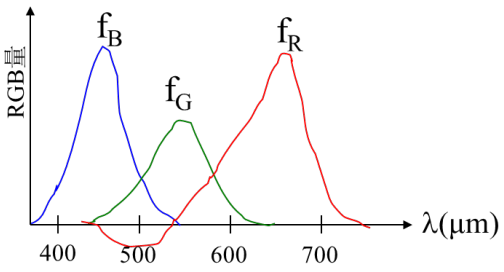
   2) 互补色：A+B=白色，则 A、B 互补

      互补色的例子：红－青、绿－品红、兰－黄

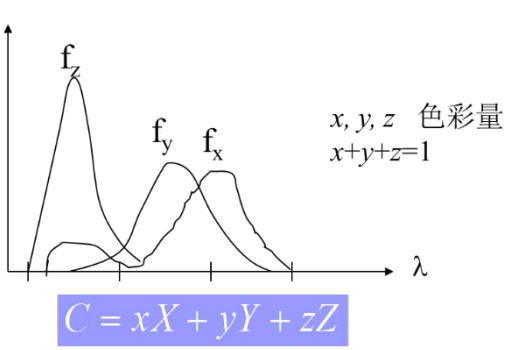
   3) 颜色匹配曲线

      $c=rR+gG+bB$

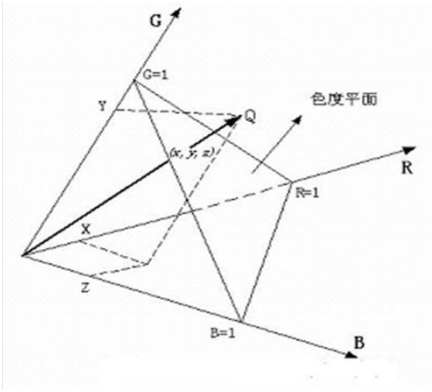
      缺陷：红绿蓝三原色系统只能表示整个光谱的一个子集，不可能生成所有的颜色，不能表示 500μm 左右的颜色。

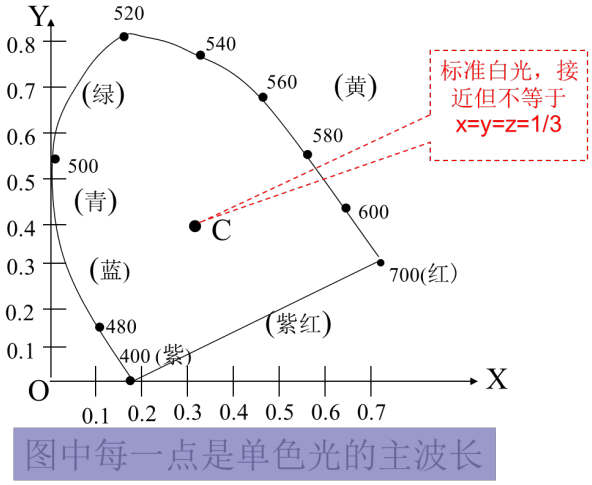


4) XYZ 颜色模型：CIE(国际照明委员会)基色



$$C = xX + yY + zZ$$





图中每一点是单色光的主波长

为了避免颜色匹配系统 RGB 为负，取三基色 XYZ，所有的可见光对应的颜色在 XZY 坐标系中组成了一个锥体。

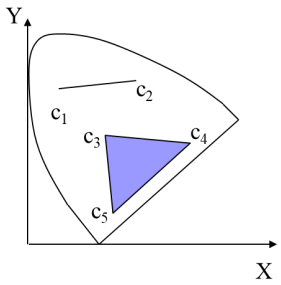
对于空间直线 OQ 上的点，X、Y、Z 三分量比例相同，表示色彩相同、亮度有差异的颜色。色度坐标：$x = \frac{X}{X+Y+Z}$；$y = \frac{Y}{X+Y+Z}$；$z = \frac{Z}{X+Y+Z}$，是原点与（X，Y，Z）构成的空间的直线 OQ 与平面 X+Y+Z=1 的交点（在用一条线上的色彩相同，亮度有差异）。但是，不可能充满整个三角形，最终形成的平面是马蹄形的，令 Z=0，，投影到 XOY 平面上，得到 CIE-XYZ 色度图。:

二基色：如果选 C1、C2 作为二基色，连线 C1C2 线上点所代表的颜色均可有适量的 C1、C2 混合而得。$c = ac1 + bc2$

三基色：三角形三点作三基色合成三角形中的颜色。【满足三基色的条件：用适当比例的这三种颜色，可以获得白色；用这三种颜色中的任意两种的组合都不能得

到第三种颜色】

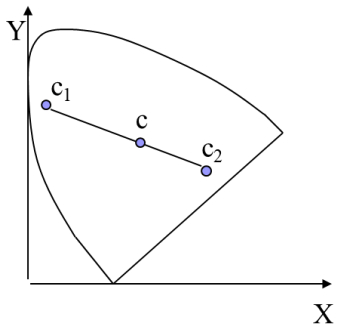
【注意：没有一个三角形能包含所有颜色---没有一个三基色组能通过加色混合生成所有颜色】

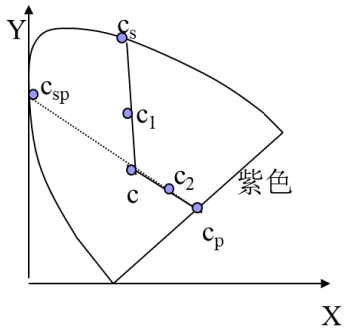


互补色： C_1、C_2 混合得白色 C， C_1,C_2 在 c 两边

颜色： C C_1 C_s （不适用 C 与紫色之间 C C2 Cp，Cp 是不存在的）

纯度： $dc1/dcs$ $(|c1c|/|csc|)$，cs 处纯度 100%

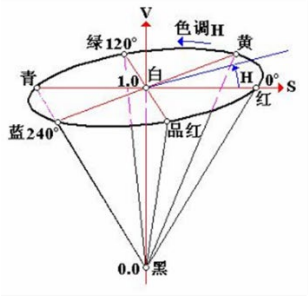


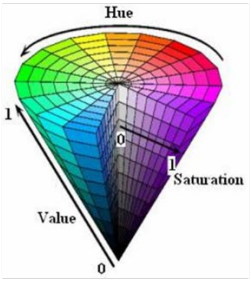


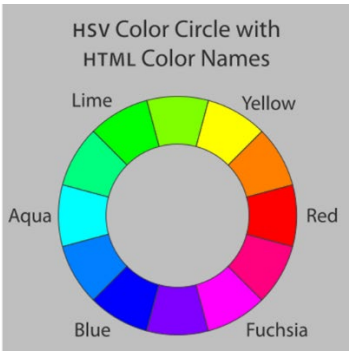
4. **颜色模型**：指某个三维颜色空间中的一个可见光子集，它包含某个颜色域的所有颜色
5. **常用的颜色模型**：
   1) 面向硬件的颜色模型
      a) RGB 模型
      b) CMY 模型（Cyan 青、Magenta 品红、Yellow 黄）
      c) YIQ 模型（Y：亮度，I、Q：色差）
   2) 面向用户的颜色模型：

      HSV 模型（H(Hue)色彩，S(Saturation)饱和度，V(Value)明度）
6. **HSV 颜色模型：**

   HSV（Hue, Saturation, Value）颜色空间的模型对应于圆柱坐标系中的一个圆锥形子集







色彩（H）和饱和度（S）构成极坐标，值（V）构成第 3 维 V 轴。

色彩（H）由绕 V 轴的旋转角给定。红色对应于角度 0°，绿色对应于角度 120°，蓝色对应于角度 240°，每一种颜色和它的补色相差 180°。

饱和度 S 取值从 0 到 1，圆锥顶面的半径为 1。

在圆锥的顶点(即原点)处，V=0，H 和 S 无定义，代表黑色。

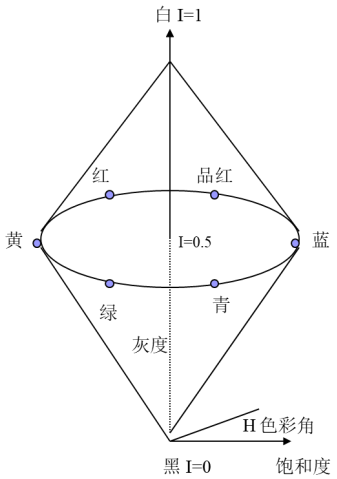
圆锥的顶面中心处 S=0，V=1，H 无定义，代表白色。

从该点到原点代表亮度渐暗的灰色，即具有不同灰度的灰色。对于这些点，S=0，H 的值无定义。

在圆锥顶面的圆周上的颜色，V=1，S=1，这种颜色是纯色。

7. **HSI 颜色模型：**

HSI 色彩空间是从人的视觉系统出发，用色调（Hue）、色饱和度（Saturation）和亮度（Intensity）来描述色彩。

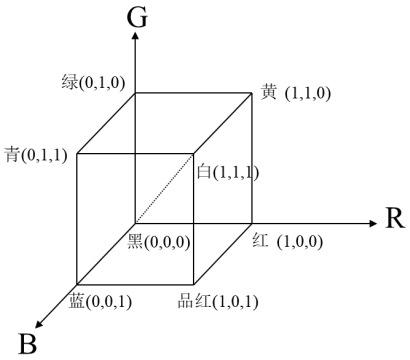


8. **RGB 颜色模型：**

三刺激理论：630μm(红) 530μm(绿) 450μm（蓝）对视网膜锥状细胞刺激最强

$$C入 = rR + gG + bB$$

RGB 立方体



显示彩色图象用 RGB 相加混色模型【加色模型】

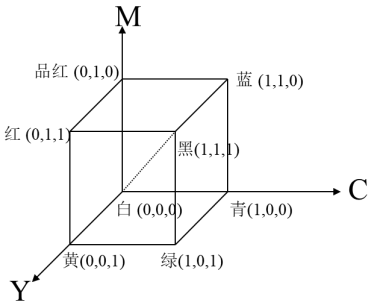
颜色＝R(红色的百分比)＋G(绿色的百分比)＋B(蓝色的百分比)

9. **CMY 颜色模型：**

三基色： Cyan（青）、Magenta（品红）、Yellow（黄）

打印彩色图象用 CMY 相减混色模型

减色模型，常用于印刷，在白纸上减色



10. **颜色模型转换：**

1) $RGB \leftrightarrow CIE\ XYZ$

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.607 & 0.174 & 0.201 \\ 0.299 & 0.587 & 0.114 \\ 0 & 0.066 & 1.117 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.910 & -0.532 & 0.288 \\ -0.985 & 1.999 & -0.028 \\ 0.058 & -0.118 & 0.898 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

2) $RGB \leftrightarrow YIQ$:

YIQ 是北美电视系统（NTSC）所采用的。Y 指亮度，I 指色彩，Q 指饱和度

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.211 & -0.523 & 0.312 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0.956 & 0.621 \\ 1 & -0.272 & -0.647 \\ -1 & -1.106 & -1.703 \end{bmatrix} \begin{bmatrix} Y \\ I \\ Q \end{bmatrix}$$

3) $RGB \leftrightarrow YUV$

YUV 是欧洲的电视系统所采用（PAL）。Y 指亮度，U 指色彩， V 指饱和度。

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.148 & -0.289 & 0.437 \\ 0.615 & -0.515 & -0.1 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1.14 \\ 1 & -0.395 & -0.581 \\ -1 & 2.032 & 0 \end{bmatrix} \begin{bmatrix} Y \\ U \\ V \end{bmatrix}$$

第九章 · 曲线曲面表示

1. **显式、隐式和参数表示：**

曲线（面）的显式表示：$y = f(x)$

曲线（面）的隐式表示：$f(x, y) = 0$

曲线（面）的参数表示：$x = x(t), y = y(t)$

【点的每一个坐标都表示成参数变量的函数。参数整个变化范围对应整条曲线，但往往只对某一部分感兴趣，通过规格化使参数的变化范围限制在[0，1]中】

显式表示特点：每一个 x 值只能对应一个 y 值；不能表示封闭或多值曲线，例如圆。

隐式表示特点：可以表示封闭或多值曲线。

非参数表示（显、隐）特点：与坐标轴相关；会出现斜率无穷大情况；非平面曲线（面）难以常系数函数表示；不便于计算和编程。

➢ 曲线的表示

1. **参数曲线定义：**

一条三维曲线的参数形式：$x = x(t), y = y(t), z = z(t), 0 \le t \le 1$

曲线上任意一点可用矢量$p(t) = [x(t), y(t), z(t)]$表示。起始点在$t = 0$处，即$p(0)$，终

点$p(1)$在$t=1$处

曲线上任意一点切矢量表示了曲线上该点的切线方向，可以表示曲线在该点处的走向。切矢量表示为：$p'(t)=[x'(t),y'(t),z'(t)]$。起始点$t=0$处切矢量，为$p'(0)$，$p'(1)$表示在终点$t=1$处切矢量。

2. **Bezier 曲线：**

在空间给定 n+1 个点 P0,P1,⋯,Pn，称下列参数曲线为 n 次 Bezier 曲线：

$$P(t)=\sum_{i=0}^{n}P_iJ_{i,n}(t),0\le t\le 1 \quad (9-1)$$

其中，$J_{i,n}(t)$是 Bernstein 基函数：

$$J_{i,n}(t)=C_n^it^i(1-t)^{n-i}=\frac{n!}{i!\,(n-i)!}t^i(1-t)^{n-i} \quad (9-2)$$

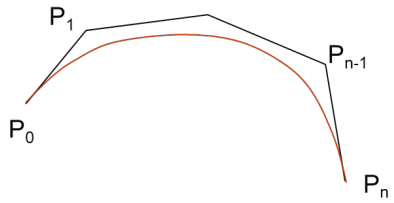
称折线为$P0P1\dots Pn$为$P(t)$的控制多边形；称点$P0,P1,\dots,Pn$为$P(t)$的控制顶点。控制多边形$P0P1\dots Pn$为$P(t)$的大致形状的勾画，$P(t)$是对$P0P1\dots Pn$的逼近。

Bezier 曲线的性质：

1) 端点的位置：$P0$和Pn是曲线$P(t)$的两个端点。$p(0)=P0,\ p(1)=Pn$

2) 端点的切线：$P(t)$在起点处与$P0P1$相切，在终点处与$Pn-1Pn$相切

$$P'(0)=n(P1-P0),p'(1)=n(P_n-P_{n-1})$$



3) 凸包性：$P(t)$位于控制顶点$P0,P1,\dots,Pn$的凸包内。

4) 几何不变性：曲线的形状仅由控制点的位置决定，与所选用的坐标系无关。

5) 交互能力：控制点的位置决定了曲线的形状，变换控制点位置，就可以改变曲线的形状。为人机交互确定曲线形状提供了手段。

Bezier 曲线的拼接：

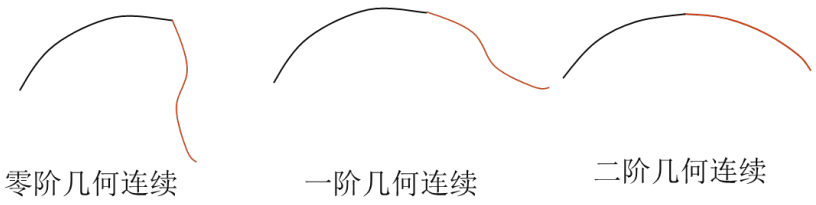
一段 Bezier 曲线常常不足以表现复杂的曲线。为了构造复杂的曲线，常用曲线拼接的方法，即用一段段的曲线首尾相连起来。

不同的问题在连接点对曲线连续性有不同的要求：

零阶几何连续：首位连在一起即可。不光滑。

一阶几何连续：在连接点处曲线方向相同。光滑，但变化趋势不同

二阶几何连续：两段曲线在连接处，不仅光滑连接，曲率也相同。



设两段 Bezier 曲线$P(t), Q(t)$首尾相接

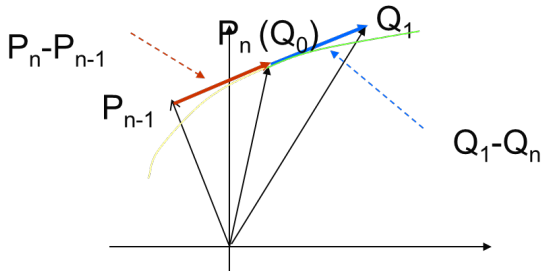
零阶几何连续就是要求：$P(1) = Q(0)$。

由 Bezier 曲线的$Q(0) = Q0,\ P(1) = Pn$，所以要使两段 Bezier 实现零阶几何连续拼接，就要求$Pn = Q0$，即前一段曲线的最后一个控制点和后一段曲线的第一个控制点重合。

一阶几何连续就是要求：$P'(1) = Q'(0)$。

由式(9-1),(9-2)可得：$P'(1) = n(P_n - P_{n-1})\ , Q'(0) = m(Q1 - Q0)$

几何意义要求：$Pn - 1\ , Pn\ (Q0)\ , Q1$点在同一条直线上，且$Pn - 1\ , Q1$分别在$Pn\ (Q0)$两边



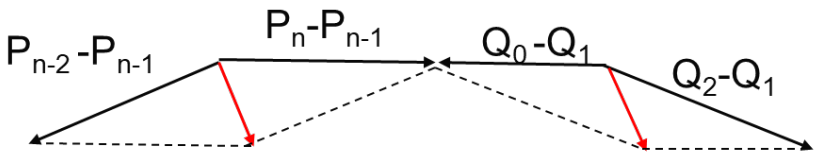
二阶几何连续就是要求：$P"(1) = Q"(0)$

由式(9-1),(9-2)可得：

$$P"(1) = n(n-1)(P_n - 2P_{n-1} + P_{n-2}) = n(n-1)(P_n - P_{n-1} + P_{n-2} - P_{n-1})$$

$$Q"(0) = m(m-1)(Q2 - 2Q1 + Q0) = m(m-1)(Q2 - Q1 + Q0 - Q1)$$

几何意义：除了对 Pn-1 , Pn (Q0) , Q1 控制点的位置有要求外，还对 Pn-2 , Q2 控制点的位置有要求。它们的位置必须使两个红色的矢量相等。



在实际应用 Bezier 曲线段绘制曲线时，常运用 3 次曲线，即 n=3。这是因为：

1) 一次曲线是直线，二次曲线是抛物线，它们表现复杂曲线的变化能力稍差；而三次曲线在一个小的局部，足够了。

2) 次数太大，数学计算量大，且容易产生累积误差，影响精度。

3) 次数太大，一段曲线的控制点太多，不利于人机交互调整曲线形状。

3 次 Bezier 曲线的计算公式：

$$P(t)=\sum_{i=0}^{3}P_iJ_{i,3}(t)=\sum_{i=0}^{3}P_i\frac{3!}{i!\,(3-i)!}t^i(1-t)^{3-i}$$

$$=P_0(1-t)^3+3P_1t(1-t)^2+3P_2t^2(1-t)+P_3t^3\quad(9-3)$$

用矩阵表示为：$p(t)=[t^3\quad t^2\quad t\quad 1]\begin{bmatrix}-1&1&-1&1\\3&-2&1&0\\-3&1&0&0\\1&0&0&0\end{bmatrix}\begin{bmatrix}P_0\\P_1\\P_2\\P_3\end{bmatrix}\quad(9-4)$

P(t)是空间点，用坐标形式表示为$[x(t),y(t),z(t)]$

由式（9-3）可知，实际坐标值可用下式分别计算：

$$x(t)=X_0(1-t)^3+3X_1t(1-t)^2+3X_2t^2(1-t)+X_3t^3$$

$$y(t)=Y_0(1-t)^3+3Y_1t(1-t)^2+3Y_2t^2(1-t)+Y_3t^3$$

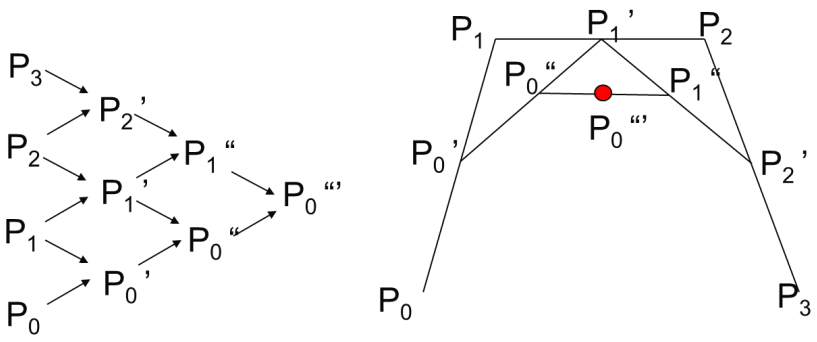
$$z(t)=Z_0(1-t)^3+3Z_1t(1-t)^2+3Z_2t^2(1-t)+Z_3t^3$$

$$0\le t\le 1\quad(9-5)$$

只要在[0,1]计算若干个 t 值，如$t=0,0.1,0.2,\dots,1$,求出相应的坐标，用直线将这些点连接起来，就构成了曲线。

若要画二维曲线，只需要计算$[x(t),y(t)]$即可。

Bezier 曲线的手工生成：

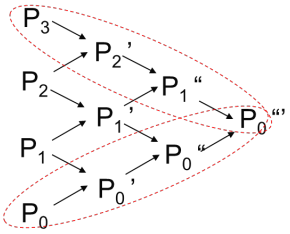


上图表示的$P0'''$是曲线上的点$P(1/2)$，每次取线段的一半；

如果每次取得不是中点，而是 1/3 点，最终得到的点是$P(1/3)$；

如果每次取得不是中点，而是 1/n 点，最终得到的点是$P(1/n)$；

$P0'''$将一段曲线分成两段，每一段成为一段新的 Bezier 曲线，其新的控制点如图所示：



3. **B 样条曲线：**

定义：设$P1, P2, \ldots, Pn(n >= k)$为给定空间的 n 个点，称下列参数曲线

$$P(t) = \sum_{i=1}^{n} P_i B_{i,k}(t), \quad t_k \le t \le t_{n+1} \quad (9-6)$$

为 k 阶或 k-1 次的样条曲线。折线 $P1, P2, \ldots, Pn$为 P(t)的控制多边形，这 n 个点为 P(t)的控制顶点。

B 样条曲线定义式与 Bezier 曲线形式类似，不同之处在于，其基函数由 Bernstein 基函数变成了 B 样条基函数$B_{i,k}(t)$

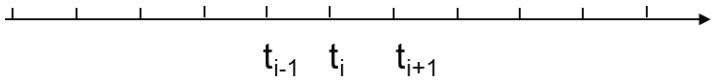
B 样条曲线基函数：

B 样条基函数是由参数 k 的递推关系所定义的：

$$B_{i,1}(t) = \begin{cases} 1, t_i \le t \le t_{i+1} \\ 0, \text{其它} \end{cases}$$

$$B_{i,k}(t) = \frac{t-t_i}{t_{i+k-1}-t_i} B_{i,k-1}(t) + \frac{t_{i+k}-t}{t_{i+k}-t_{i+1}} B_{i+1,k-1}(t) \qquad -\infty < t < +\infty$$

其中，$T=\{ti\}, i=0,\pm1,\pm2,\ldots\ldots$是对参数轴 t 的一个划分：



如果划分是等距离的，即对于任意 i, $t_{i+1}-t_i=(i+1)-i=1$，形成的是均匀 B 样条曲线。我们只讨论均匀 B 样条曲线。

B 样条基函数是递推定义：$B_{i,1}(t)\rightarrow B_{i,2}(t)\rightarrow B_{i,3}(t)\rightarrow B_{i,4}(t)\rightarrow\cdots$

B 样条基函数的次数：

k=1 时，$B_{i,k}(t)$常数 1，零次函数；

k=2 时，$B_{i,k}(t)$为一次函数；

k=3 时，$B_{i,k}(t)$为二次函数；

k=4 时，$B_{i,k}(t)$为三次函数。

B 样条基函数分析：局部性

$B_{i,1}(t)$的定义域为$[ti,ti+1)$

$B_{i,2}(t)$由两段$B_{i,1}(t)$，$B_{i+1,1}(t)$组成，定义域为$[ti,ti+1)\cup[ti+1,ti+2)=[ti,ti+2)$

$B_{i,3}(t)$的定义域为$[ti,ti+3)$

$B_{i,4}(t)$的定义域为$[ti,ti+4)$

意味着一个控制点 Pi 对整个曲线的影响只局限在 k 个分割区域$[ti,ti+k)$中

$$P(t)=P_1B_{1,k}(t)+P_2B_{2,k}(t)+\cdots+P_iB_{i,k}(t)+\cdots+P_nB_{n,k}(t)$$

反之，一段定义在区间$[ti,ti+1)$上的 B 样条曲线 Pi(t)只由 k 个控制点决定$P_{i-k+1},\ldots,P_{i-1},P_i$，整个 B 样条曲线就是这样一段一段绘制完成的，只要能解决每一段定义在区间$[ti,ti+1)$上的 B 样条曲线 Pi(t)的绘制就能绘制整段曲线。整个曲线是由$n-k+1$段曲线组成的。

B 样条曲线段的计算公式推导：

以最简单的 k=2 情况为例，说明如何由定义式推出曲线段的计算公式：

k=2，一段曲线$B_{i,2}(t)$只有两个控制点

$$P(t)=\sum_{i=1}^{2}P_iB_{i,2}(t)=P_1B_{1,2}(t)+P_2B_{2,2}(t)\ ,\ t_2\le t\le t_3$$

其中：$B_{1,2}(t)=\frac{t-t_1}{t_2-t_1}B_{1,1}(t)+\frac{t_3-t}{t_3-t_2}B_{2,1}(t),-\infty<t<+\infty$

而：$B_{1,1}(t)=1,t_1\le t\le t_2$ ； $B_{2,1}(t)=1,t_2\le t\le t_3$

故：$B_{1,2}(t)=\begin{cases}\frac{t-t_1}{t_2-t_1},t_1\le t\le t_2\\ \frac{t_3-t}{t_3-t_2},t_2\le t\le t_3\end{cases}$ ； $B_{2,2}(t)=\begin{cases}\frac{t-t_2}{t_3-t_2},t_2\le t\le t_3\\ \frac{t_4-t}{t_4-t_3},t_3\le t\le t_4\end{cases}$

在定义域$t2\le t<t3$中，$P(t)=P_1\frac{t_3-t}{t_3-t_2}+P_2\frac{t-t_2}{t_3-t_2}=P_1(3-t)+P_2(t-2)$

令$t'=t-t2$，$P(t')=P_1(1-t')+P_2t'=[1-t'\quad t']\begin{bmatrix}P_1\\P_2\end{bmatrix}=[t'\quad 1]\begin{bmatrix}-1&1\\1&0\end{bmatrix}\begin{bmatrix}P_1\\P_2\end{bmatrix}(0\le t'<1)$

用熟悉的 t 参数来表示$k=2$时的曲线表示为：$P_i(t)=[t\quad 1]\begin{bmatrix}-1&1\\1&0\end{bmatrix}\begin{bmatrix}P_{i-1}\\P_i\end{bmatrix},0\le t\le 1$

同理：$k=3$、4时的第 i 段曲线的矩阵表示为：

$$k=3:\ P_i(t)=\frac{1}{2}[t^2\quad t\quad 1]\begin{bmatrix}1&-2&1\\-2&2&0\\1&1&0\end{bmatrix}\begin{bmatrix}P_{i-2}\\P_{i-1}\\P_i\end{bmatrix},0\le t\le 1$$

$$k=4:\ P_i(t)=\frac{1}{6}[t^3\quad t^2\quad t\quad 1]\begin{bmatrix}-1&3&-3&1\\3&-6&3&0\\-3&0&3&0\\1&4&1&0\end{bmatrix}\begin{bmatrix}P_{i-3}\\P_{i-2}\\P_{i-1}\\P_i\end{bmatrix},0\le t\le 1$$

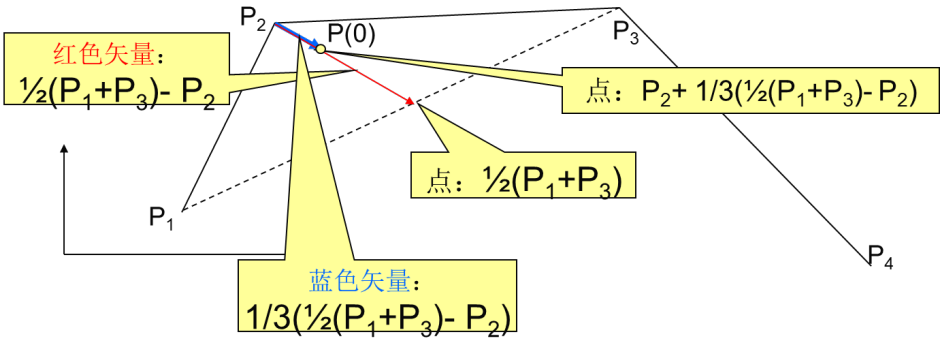
B 样条曲线的性质（几何性质）：

以一段 3 次曲线(k=4)为例，说明 B 样条曲线的性质。P(t)的控制点为 P_1,P_2,P_3,P_4

1) 端点：P(t)的起点为 P(0) ，终点为 P(1)，不在两端的控制点上。

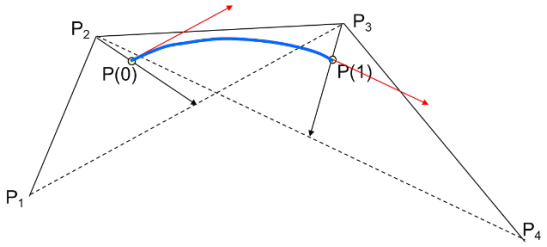
$$P(0)=\frac{1}{6}(P1+4P2+P3)=P2+\frac{1}{3}\left(\frac{1}{2}(P1+P3)-P2\right)$$

$$P(1)=\frac{1}{6}(P2+4P3+P4)=P3+\frac{1}{3}\left(\frac{1}{2}(P2+P4)-P3\right)$$



2) 端点的切线:$P'(0)$，$P'(1)$:

$$P'(0) = 1/2(P3 - P1), P'(1) = 1/2(P4 - P2)$$



3) 凸包性：曲线段在控制点所围成的凸包内

4) 连续性：第一个曲线段的控制点为 P1,P2,P3,P4,第二个曲线段的控制点为 P2,P3,P4,P5,相邻的两个曲线段有三个共同的控制点，即决定第一段终点的控制点与决定第二段起点的控制点完全一样:

$$P(1) = Q(0) = 1/6(P2 + 4P3 + P4)\ ;\ P'(1) = Q'(0) = 1/2(P4 - P2)$$

第一段终点的位置和方向与第二段起点的位置和方向完全一致，因此相邻的曲线段自动连接。可以证明，曲线在连接处具有 k-1 阶连续。这是 B 样条曲线优于 Bezier 曲线的地方，它能自动地平滑连接。

4. **三次 Hermite 曲线**:

一条三次参数曲线的代数形式是:

$$P(t) = a_3t^3 + a_2t^1 + a_1t + a_0, t \in [0,1]$$

四个系数 a3、a2 、a1、a0 唯一地确定了曲线的形状和位置。四个系数需要四个已知条件来确定，这种根据四个已知条件来确定的一条三次参数曲线称为三次 Hermite 曲线。

常用的两种四个已知条件可以取:

1) 两个端点$P(0)$、$P(1)$和对应的切矢量$P'(0)$、$P'(1)$;

2) 曲线上均匀分布的四个点$P(0)$、$P(1/3)$、$P(2/3)$、$P(1)$。

推导出由均匀分布的四个点$P(0)$、$P\left(\frac{1}{3}\right)$、$P\left(\frac{2}{3}\right)$、$P(1)$决定的三次 Hermite 曲线:

$$P(0) = a0;\ P\left(\frac{1}{3}\right) = \frac{1}{27a3} + \frac{1}{9a2} + \frac{1}{3a1} + a0;$$

$$P(2/3) = 8/27a3 + 4/9a2 + 1/3a1 + a0;\ P(1) = a3 + a2 + a1 + a0$$

解得四个参数为:

$$a3 = -9/2P(0) + 27/2P(1/3) - 27/2P(2/3) + 9/2P(1)$$

$$a2 = 9P(0) - 45/2P(1/3) + 18P(2/3) - 9/2P(1)$$

$$a1 = -11/2P(0) + 9P(1/3) - 9/2P(2/3) + P(1)$$

$$a0 = P(0)$$

用矩阵表示即为:

$$P(t) = [t^3 \quad t^2 \quad t \quad 1]\begin{bmatrix} -\frac{9}{2} & \frac{27}{2} & -\frac{27}{2} & \frac{9}{2} \\ 9 & -\frac{45}{2} & 18 & -\frac{9}{2} \\ -\frac{11}{2} & 9 & -\frac{9}{2} & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} P(0) \\ P(\frac{1}{3}) \\ P(\frac{2}{3}) \\ P(1) \end{bmatrix}$$

Hermite 曲线的性质:

1) Hermite 曲线的控制点直接位于曲线之上，这种控制点称为型值点;

2) 三次 Hermite 曲线的起点和终点分别位于第一和第四个控制点上，控制点应该分成每四个一组，每一组形成一段曲线。为了使每个曲线段连接起来，相邻两组的控制点应有一个控制点是重复的，即前一组的最后一个控制点同时也是后一组的第一个控制点。

➢ 曲面的表示

1. 曲面和曲线一样，也是用参数的形式表示。不同的是，曲面参数是二维的，用(u, v)表示。曲面和曲线类似，是用分片表示的（曲线是分段）。整个曲面是由一系列的分片拼接而成的。

2. **Bezier 曲面：**

一个三次 Bezier 曲面片的控制点有 4x4 个，对应的代数形式为：

$$P(u,v)=\sum_{i=0}^{3}\sum_{j=0}^{3}P_{ij}J_{i,3}(u)J_{j,3}(v)$$

用矩阵表示为：

$$P(u,v)=[u^3 \quad u^2 \quad u \quad 1]\begin{bmatrix}-1&1&-1&1\\3&-2&1&0\\-3&1&0&0\\1&0&0&0\end{bmatrix}\begin{bmatrix}P_{00}&P_{01}&P_{02}&P_{03}\\P_{10}&P_{11}&P_{12}&P_{13}\\P_{20}&P_{21}&P_{22}&P_{23}\\P_{30}&P_{31}&P_{32}&P_{33}\end{bmatrix}\begin{bmatrix}-1&3&-3&1\\1&-2&1&0\\-1&1&0&0\\1&0&0&0\end{bmatrix}\begin{bmatrix}v^3\\v^2\\v\\1\end{bmatrix}$$

3. **B 样条曲面：**

一个三次 B 样条曲面片的控制点有 4x4 个

一个三次 B 样条曲面片的代数形式是：

$$P(u,v)=\sum_{i=0}^{3}\sum_{j=0}^{3}P_{ij}B_{i,3}(u)B_{j,3}(v)$$

用矩阵表示为：

$$P(u,v)=\frac{1}{36}[u^3 \quad u^2 \quad u \quad 1]\begin{bmatrix}-1&3&-3&1\\3&-6&3&0\\-3&0&3&0\\1&4&1&0\end{bmatrix}\begin{bmatrix}P_{00}&P_{01}&P_{02}&P_{03}\\P_{10}&P_{11}&P_{12}&P_{13}\\P_{20}&P_{21}&P_{22}&P_{23}\\P_{30}&P_{31}&P_{32}&P_{33}\end{bmatrix}\begin{bmatrix}-1&3&-3&1\\3&-6&0&4\\-3&3&3&1\\1&0&0&0\end{bmatrix}\begin{bmatrix}v^3\\v^2\\v\\1\end{bmatrix}$$

4. **Hermite 曲面：**

一个三次 Hermite 曲面片的控制点有 4x4 个

一个三次 Hermite 曲面片的代数形式是：

$$P(u,v)=\sum_{i=0}^{3}\sum_{j=0}^{3}P_{ij}H_{i,3}(u)H_{j,3}(v)$$

用矩阵表示：

$$P(u,v)=\frac{1}{4}[u^3 \quad u^2 \quad u \quad 1]\begin{bmatrix}-9&27&-27&9\\18&-45&36&-9\\-11&18&-9&2\\2&0&0&0\end{bmatrix}\begin{bmatrix}P_{00}&P_{01}&P_{02}&P_{03}\\P_{10}&P_{11}&P_{12}&P_{13}\\P_{20}&P_{21}&P_{22}&P_{23}\\P_{30}&P_{31}&P_{32}&P_{33}\end{bmatrix}\begin{bmatrix}-9&18&-11&2\\27&-45&18&0\\-27&36&-9&0\\9&-9&2&0\end{bmatrix}\begin{bmatrix}v^3\\v^2\\v\\1\end{bmatrix}$$

5. 曲面$P(u,v)$上的任意点可以通过给定参数u,v $(0<=u,v<=1)$、计算$P(u,v)$来得到。

曲面$P(u,v)$上的曲线可以通过固定 u 或 v 得到，例如 $P(0,v)$， $P(1/2,v)$， $P(1,v)$，

$P(u,0)$

6. 一个曲面常用几条曲面上的曲线来表示。

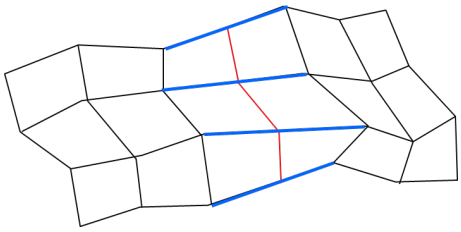
7. **曲面片的连接：**

1) **Bezier 曲面片的连接**:

两个 Bezier 曲面片分别由Pij、$Qij(i,j=1,\dots,4)$控制点确定。

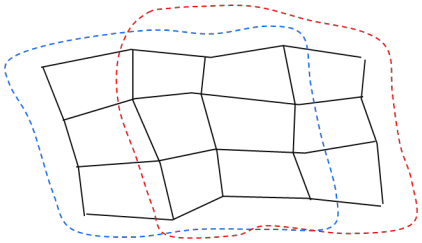
如果要求两个曲面片相连，则要求控制点$Pi4 = Qi1(i=1,\dots,4)$;

如果要求两个曲面片在连接处一阶平滑，则要求控制点$Pi3, Pi4\ (Qi1), Qi2$在一条直线上$(i=1,\dots,4)$ 。



2) **B 样条曲面的连接：**在两个曲面连接处自动实现光滑连接

4x5 个控制点$Pij\ (i=1,\dots,4\ ;\ j=1,\dots,5\ )$决定两个相邻曲面的形状，其中$Pij\ (i=1,\dots,4\ ;\ j=1,\dots,4\ )$决定一个曲面的形状，$Pij\ (i=1,\dots,4\ ;\ j=2,\dots,3\ )$决定另一个曲面的形状。两个相邻曲面拥有 12 个共同的控制点$Pij\ (i=1,\dots,4\ ;\ j=2,\dots,4\ )$



3) **Hermite 曲面的连接**:

两个 Hermite 曲面片分别由Pij、$Qij(i,j=1,\dots,4)$控制点确定。如果要求两个曲面片相连，则要求控制点$Pi4 = Qi1(i=1,\dots,4)$；如果要求两个曲面片在连接处平滑，则要调整控制点。

8.